Control over variability in nonvolatile hafnium-oxide resistive-switching memory based on modeling of the switching processes

Brian Jerad Butcher

University at Albany, State University of New York, brianjbutcher@gmail.com

The University at Albany community has made this article openly available. Please share how this access benefits you.

Follow this and additional works at: https://scholarsarchive.library.albany.edu/legacy-etd

Part of the Nanoscience and Nanotechnology Commons

Recommended Citation

https://scholarsarchive.library.albany.edu/legacy-etd/844

This Dissertation is brought to you for free and open access by the The Graduate School at Scholars Archive. It has been accepted for inclusion in Legacy Theses & Dissertations (2009 - 2024) by an authorized administrator of Scholars Archive. Please see Terms of Use. For more information, please contact scholarsarchive@albany.edu.
CONTROL OVER VARIABILITY IN NONVOLATILE HAFNIUM-OXIDE RESISTIVE-SWITCHING MEMORY BASED ON MODELING OF THE SWITCHING PROCESSES

by

Brian Jerad Butcher

A Dissertation
Submitted to the University at Albany, State University of New York
in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy

College of Nanoscale Science and Engineering
2013
CONTROL OVER VARIABILITY IN NONVOLATILE HAFNIUM-OXIDE RESISTIVE-SWITCHING MEMORY BASED ON MODELING OF THE SWITCHING PROCESSES

by

Brian Jerad Butcher

COPYRIGHT 2013
To my Father, Ron Butcher, for sparking my creative interest in science and engineering by teaching me how to make luminous paints and homemade gun-power in the garage as a boy.

To my ‘Nick-Dad’, Robert Sutter, for teaching me the value of stopping-to-think, plan, and appreciate opportunities before delving aimlessly into working-hard.

And to my Wife, Elizabeth Butcher, for your support, encouragement, and most of all your love.
ABSTRACT

Resistive random access memory (ReRAM) technology presents an attractive option for embedded non-volatile (NV) memory systems if its variability (cycle-to-cycle and device-to-device) can be controlled. This dissertation has focused on investigations to identify key mechanisms and parameters which dominate ReRAM variability, and the development of subsequent experimental and simulation-based tools to address this variability. The first component of these efforts entailed identification of the modern-day non-volatile memory technological gaps that have driven the operational requirements and challenges for resistive memory as an emerging NV memory.

Initial research confirmed the critical requirement of a sub-stoichiometric (HfO$_{2-x}$) dielectric regarding the enablement of stable switching and suggested a defect-driven mechanism, which is discussed in detail. Preliminary experimental work was focused on the fabrication of a durable current-limiting (1T1R) testing structure; which was utilized to enable ReRAM device characterization, reduce unwanted parasitic capacitances, and overshoot-current. Initial electrical and physical characterization confirmed a filamentary based (defect-driven) mechanism based on ReRAM scalability-trends (in device sizes ranging from 50x50nm$^2$ to 7x7µm$^2$). Physical analysis (AFM, TEM and EELS) verified a ‘dominant-filament mechanism’ in transmission-metal–oxide (specifically HfO$_{2-x}$) based ReRAM.

A novel characterization and analysis protocol for key electrical parameters affecting filament formation for HfO$_{2-x}$-based ReRAMs was developed, focusing on the roles of current, voltage, and temperature. This protocol included characterization of the high-resistive-state (HRS) dependence on the maximum FORMING current (seen during 1st RESET $I_{\text{max}}$) and the characterization of low-power endurance. This characterization protocol was employed to investigate and develop an approach for ReRAM filament
formation at elevated temperatures (hot FORMING) to improve filament formation control and reduce variability. This method subsequently achieved low-power ReRAM switching, while maintaining a high on/off ratio. By drastically reducing time-to-FORMING, this hot FORMING method enabled filament-formation using low constant-voltage-biases, which was shown to increase resistance of the conductive filament, lower operational current, increase memory window and improve device-to-device uniformity. This novel hot-FORMING method underscored the importance of reducing the FORMING voltage ($V_F$) as to create switchable, low-required-power, conductive-filaments (CFs).

To enable further optimization of ReRAM performance from the standpoint of both device structural properties and operation conditions, a novel 3-dimensional physics-based simulation-model of the inherently stochastic and difficult-to-control ReRAM filament FORMING process was formulated using a statistical kinetic Monte-Carlo method. This simulation framework was constructed to elucidate the physical processes responsible for electrical (FORMING, RESET, and SET) characteristics by investigating the relationships between current, voltage, temperature, filament formation and variability. Specifically, this simulative investigation was carried out in terms of time-dependent-dielectric-breakdown: Hf-O bond-breakage i.e. defect generation, oxygen-ion diffusion, and vacancy-oxygen recombination. The simulation results well-reproduced the experimental trends observed for the conductivity of the post-FORMING low resistance state under different FORMING conditions. Simulation results also revealed that the distribution of the oxygen ions in the surrounding oxide during FORMING as well as local filament temperature and electrical field all affected the filament stability. This simulation model also provided, for the first time, a consistent explanation for all reported critical ReRAM features including the critical role of O-deficiency (sub-stoichiometry) needed for repeatable switching, the relation between switching characteristics and dielectric morphology, the advantages of inducing
oxygen deficiency by using the oxygen gettering process, and the kinetics of \textit{RESET} gap formation. The model also successfully reproduced, with no fitting parameters, the observed LRS and HRS trends with respect to dielectric stoichiometry, \textit{FORMING} voltage, and amplitude/duration of the \textit{RESET} input pulse.

This work provides an encompassing view of the emerging fab-friendly, HfO$_{2-x}$-based, ReRAM-technology. By combining detailed physical and electrical analysis with modeling simulations, this work connects the properties of current, voltage, temperature, and dielectric (sub-) stoichiometry with stable filament formation. Key guidelines for ‘defect-engineering’ and control over the naturally-stochastic ionic-based ReRAM systems are conveyed which can be used with regard to expediting low-cost embedded NV ReRAM applications.
ACKNOWLEDGMENTS

Firstly, I need to thank Wei Wang (my 1st research advisor at CNSE) for giving me the opportunity to work at SEMATECH as an intern during my PhD; it was a very rewarding and valuable experience. A very well desired thank you is also in order for Dr. Nathaniel Cady for his honest advice and guidance: he is a great professor who truly seems to care. I would like to acknowledge my technical advisors, Gennadi Berserker (SEMATECH-Fellow) and David Gilmer (SEMATECH), for their patience and invaluable guidance through this process. I am also very grateful for the follow people I meet while working at SEMATECH as they provided meaningful conversations (or diners) that have helped me through this Ph.D. process: Chadwin Young, Paul Kirsch, Hok-Young Park, Bill Taylor, PY Hung, Ken Matthews and Dmitry Velsker.

To my academic advisor Dr. Robert Geer I am also very grateful for keeping me on task and for funding my research and conference trips to Italy. I am very grateful for the MATLAB training I received in Italy at the University of Modena and Reggio Emilia from Luca Vandelli (Vandalo) under Professor Luca Larcher.

I would also like to thank Edward Cupoli and Laura Schultz for teaching me business plan logistics and entrepreneurship fundamentals at the start of my Ph.D.

I am thankful for the friends I have made both inside and outside the classroom/office/town: Isaac Lund, Charlie Settens, Kevin Ryan, Fernando Gomez-Baquero, Ben Backes, Jimmy Williams, Wesley Notling, Robin B Jacobs-Gedrim, David Epstein, and Abraham Hmiel.

And finally, I need to thank ASUS, Intel and Corsair Memory for enabling me to assemble ‘One-Mother-Beast-of-a-Computer’: ASUS motherboard P9X79PRO, Overclocked (4.6GHz) Intel 6-Core i7-3930K CPU, Corsair radiator cooling kit, 64GB of RAM, and a 2TB hard-drive. This work would not have been possible without this computer.
# Table of Contents

CHAPTER 1: Introduction................................................................................................................. 1
  1.1 Motivation: The Modern-Day Demand for Easily Accessible Information...................... 1
  1.2 Clarifying the Differences between Modern-Day Random-Access and Read-Only Memory... 1
  1.3 Nonvolatile Memory (NVM) Definitions.............................................................................. 2
  1.4 NVM Technology: Where does ReRAM fit-in?................................................................. 3
  1.5 Resistive-Switching Memory: Well Positioned as an Embedded Memory?....................... 8
    1.5.1 Known Features of ReRAM......................................................................................... 9
    1.5.2 Approaching the Problem in More Detail: Needed Specifications............................. 14
  1.6 Document Organization...................................................................................................... 18

CHAPTER 2: Background and Preliminary Work............................................................................. 20
  2.1 ReRAM Taxonomy Background and Preliminary Physical Analysis of Filament................. 20
    2.1.1 Background: Brief History....................................................................................... 20
    2.1.2 Background: RRAM Switching Materials: Transition-metal-oxides (TMOs).............. 21
    2.1.3 Background: Atomic-Level Mechanisms..................................................................... 22
    2.1.4 Background: Distinguishing Filamentary Switching and Area Distributed Switching ............................................................................................................................................................................. 24
    2.1.5 Preliminary Physical Analysis: Spectroscopy Used to Define Filament Cross-Sections ............................................................................................................................................................................. 27
    2.1.6 Preliminary Definitions: Defining TMO-ReRAM in terms of Filament Formation ......... 30
  2.2 Background: Defining the Benefits of Hafnium-Oxide as a RRAM (TMO) Dielectric ......... 32
  2.3 Preliminary Work: Critical Requirement of Sub-Stoichiometric (HfOx) Dielectric............. 33
  2.4 Preliminary (HfOx) Electrical Analysis............................................................................... 35
    2.4.1 One-Transistor, One-Resistor (1T1R) Testing Structure: Limiting Overshoot-Current
         and Parasitic Capacitances............................................................................................... 35
    2.4.2 Scaling Trends with Material Layer Thickness and Device Area.................................. 44
  2.5 Preliminary Theoretical Analysis......................................................................................... 50
    2.5.1 Introduction to the Dielectric-Breakdown-based RRAM Model................................. 50
    2.5.2 Theoretical FORMING/RESET/SET Switching Process Overview......................... 51
    2.5.3 Initial Leakage Current: Trap Assisted Tunneling...................................................... 56
  2.6 Discussion and Summary.................................................................................................... 59

CHAPTER 3: Filament-Formation: Characterization and Control..................................................... 60
  3.1 Introduction....................................................................................................................... 60
  3.2 Experimental.................................................................................................................... 62
3.3 Parameters affecting Filament Formation and the Low-Resistance (ON) State: Current, Voltage Sweep Rate, Voltage and Temperature ........................................................................................................ 65

3.4 High-Resistance (OFF) State Dependence on Max FORMING Current ........................................ 70

3.5 Summarizing Filament Characterization Traits .................................................................................. 73

3.6 Hot FORMING Method to Improve Memory Window and Uniformity of Low-Power HfO_{2-x} based ReRAMs .................................................................................................................. 75

3.7 SET and RESET Characterization: Low-Power Endurance Performance .................................. 83
    3.7.1 Endurance testing: dc at various temperatures ..................................................................... 83
    3.7.2 Endurance testing (ac characterization) at 150°C and room temperature ...................... 87

3.4 Discussion .................................................................................................................................. 89

CHAPTER 4: Modeling ReRAM FORMING and (RESET/SET) Switching .......................................... 91

4.1 Introduction ....................................................................................................................................... 91

4.2 Detailed Description of the Model and Simulation Approach ......................................................... 95
    4.2.1 Model Description: Initial Input Conditions ..................................................................... 95
    4.2.2 Model Description: Kinetics .............................................................................................. 97
    4.3.1 Determining the Resistivity of Sub-Stoichiometric Hafnium-Oxide ................................. 100

4.3 Modeling the Effects of Different FORMING Conditions on Conductive Filament Stability 102
    4.3.1 Simulated Filament Properties vs. Different FORMING Conditions: Comparing Different V_f and Temp. Conditions on vs. Oxygen/Vacancy Distributions .......... 102
    4.3.2 Resistance and Cross-sections of Conductive Filament .................................................. 108
    4.3.3 Evaluating the Filament Stability under Low Current Compliance ................................ 109
    4.3.4 Discussion of Filament Stability Results ............................................................................ 110

4.4 Connecting Physical and Electrical Properties of Hafnia-based ReRAM .................................. 112
    4.4.1 Effect of Oxide Sub-stoichiometry on Oxygen Ion Movement after FORMING .......... 112
    4.4.2 Effect of Pulse Amplitude/Duration on RESET Gap Formation .................................. 118
    4.4.3 Effect of Oxide Sub-Stoichiometry on RESET Gap Formation .................................. 121
    4.4.4 Discussion ......................................................................................................................... 125

4.5 Summary of Modeling Results ........................................................................................................ 126

CHAPTER 5: Conclusions and Future Directions ................................................................................... 128

5.1 Summary and Conclusions .............................................................................................................. 128

5.2 Suggested Future Directions ........................................................................................................ 130

APPENDIX 1: Other NVM Emerging Technologies (PCM, STT-MRAM, CNT-RAM) .................. 132

APPENDIX 2: Integrated 1T1R – Processing Steps ........................................................................ 136

APPENDIX 3: Description of Simulating 3D Grains/Grain-Boundaries .......................................... 159

APPENDIX 4: Executing Numerical Calculations in MATLAB ...................................................... 164

APPENDIX 5: Matlab Code .................................................................................................................... 166
CHAPTER 1: Introduction

1.1 Motivation: The Modern-Day Demand for Easily Accessible Information

In today's modern society, many people consciously or sub-consciously thrive on the stress of constant fixes of information [1]. Depending on whom you ask, a constant stream of easily accessible data (via cell phones, tablets, or laptops) empowers people to 'accelerate-their-success' or just 'prevent themselves from boredom'. Nevertheless this constant streaming of data requires an ever-increasing demand for copious-amounts of data storage which is faster, scalable, nonvolatile, cheap-to-manufacture, and easy to embed into our favorite hand-held devices. In the semiconductor industry the rising requirements for reducing-power-consumption and convenience-of-use have focused research efforts toward the technological development of nonvolatile memories. The motivation of this research is to provide a meaningful investigation of an emerging nonvolatile memory technology that could meet all these requirements pending further development: Resistive random access memory (ReRAM).

1.2 Clarifying the Differences between Modern-Day Random-Access and Read-Only Memory

Figure 1-1 shows the taxonomy of semiconductor memory, structured into volatile and nonvolatile memories. Before delving into the details of nonvolatile memory and resistive-switching memory it is important to clearly distinguish between random-access-memory (RAM) and read-only memory (ROM). RAM, in its simplest definition, has an accessible x-y-z address for each cell without disturbing other cells which allows for selective access to each memory unit and distinguishes it from other serial access (ROM) memories such as magnetic-tape memory [2]. However as ROM technologies have matured, this fundamentally distinguishing definition becomes ambiguous as modern day ROM now has selective (random) access capability, and the read processes for both types of memory
are identical. And even though RAM is called read-write memory, ROM does have some limited rewriting capabilities. Therefore, the main difference between modern ROM and RAM is the ease and frequency of erasing and programming with RAM being the easiest and most frequent, respectively [2].

Figure 1-1: Semiconductor memory taxonomy.

1.3 Nonvolatile Memory (NVM) Definitions

Mask-ROM (Mask-programmed-ROM)

EPROM (UV Erasable Programmable ROM)

OTP (One Time Programmable EPROM)

EEPROM (Electrically Erasable and Programmable ROM)

Flash/SONOS (EEPROM based Floating Gate Technology Either ROM or RAM)

ReRAM (Either Filament or Non-Filament Based)

STT-MRAM (Spin-Transfer Torque Magnetic RAM)
**PCM (Phase Change Memory)**

**NRAM (Carbon Nanotube Based Memory)**

**Magnetic Tape Storage:** Thin magnetizable coating on a long, narrow strip of plastic film

**HDD (Hard Disk Drive):** mechanical RAM using rapidly rotating discs with magnetic heads

**ODD (Optical Disk Drive):** mechanical RAM that reads Compact (Laser) Disks

**1.4 NVM Technology: Where does ReRAM fit-in?**

Nonvolatile memories are low power and more robust than other types of memory. Theoretically, they are defined as any bistable (on/off; 1 or 0) device or element that retains its state when disconnected from a power source [3]. On the other hand, volatile memory will lose stored data, in seconds, when the power is terminated and, therefore, more commonly used for temporary storage. Examples include Dynamic-RAM (DRAM) and Static-RAM (SRAM) technologies [2, 3]. Many materials and/or nanoelectronic device combinations have been evaluated for nonvolatile, bistable technologies. Thus, NVM is a rapidly developing field of study, and filled with a vast amount of different technology either developed (Mask-ROM, PROM, EPROM, EEPROM, Flash/SONOS, mechanical Magnetic Tape Storage, mechanical HDD and ODDS) or under preliminary development (ReRAM, PCM, STT-MRAM, NRAM). Semiconductor based nonvolatile memory in the recent past was typically limited to ROM and used only for long-term storage until the recent development of transistor-based Flash which is random-access (RAM). And as NVM continues to mature (becoming faster, scalable, cheaper-to-manufacture, and easy to embed) the semiconductor industry recognizes the following technical performance metric categories for nonvolatile memory [2, 3]:

1) **Speed:** in terms of accessing storage data, programing and eraser time.

2) **Retention:** measures how long the memory bit can retain its programmed state.
3) **Endurance:** number of program and erase cycles before memory device failure

4) **Power Consumption:** needed to program, erase, and access memory bit data.

5) **Power supply voltages:** needed to program, erase and access memory data.

6) **Memory cell size**

7) **Scaling Properties**

**Wide Field of Applications for Nonvolatile Memories**

**Consumers**
- Computers, Tablets, Phones
- Appliances
- Automotive, transportation

**Corporate**
- Computers
- Energy, Infrastructure: Smart meters
- Storage, Networking: Data centers
- Medical

*Figure 1-2: Applications for non-volatile memory technology*

NVM has a wide-field of applications for both consumer and corporate products. As seen in Fig. 1-2, the consumer market consists of computers, tablets, phones, appliances and transportation. The corporate market also includes computers but ranges from energy storage, smart meters, data centers, networking and medical devices [4]. As in most markets, new-technologies disrupt prior applications (Fig. 1-3). Figure 1-3 describes the relative NVM product-age versus the total-volume-amount-of-applications-in-production. This figure indicates that as a given NVM technology matures it is used in fewer and fewer
products as consumer demand declines. Solid-state hard-drives, and storage area networks are two recent entries in the NVM market and as consumer demand rises, production will increase, [4] thus reinforcing the need for more advanced, more compact nonvolatile data storage.

Figure 1-3: Life cycle curve of NVM products.

Figure 1-4: Timeline in which each new NVM entered the market, showing the relative density increase and added feature of each new product.
The NVM timeline, as seen in Fig. 1-4, describes which new feature(s), excluding cost-reduction, set each new technology apart from its predecessor and enabled increases in overall storage density. Before discussing ReRAM potential it is important to understand which types of NVMs are still around today and for what reasons.

**Mask-programmed ROM** was the first type of integrated-circuit memory and is based on a patterned photolithographic process [2-5]. The total amount of code storage memory (too small for broader data storage) is fixed by the manufacturer and not re-programmable by the end-user. Per-bit mask-programmed-ROM is the most compact type of memory. And since the cost of an integrated circuit (IC) depends on its size, mask-programmed ROM can be significantly cheaper than any other type of memory (provided overall production is large). Yet, the required one-time photolithographic pattern development has a long turnaround time, and very high costs (in terms of development). Furthermore, if any design errors are discovered after development, the costly mask may not be usable.

**Programmable read-only memory (PROM/OTP)** like mask-programmable ROM, provides on-chip code storage, but has the added feature of a one-time, on-chip programming (thus the acronym OTP). This technology is still around today, and it allows manufacturers to keep a supply of blank PROMs in stock for on-demand programming to avoid large overstock of one particular type of coded memory. PROM can be programmed at the wafer/development, final-test, or system level (example: car key fobs with embedded chips, and video game consoles). PROM uses a programmable ‘fuse/anti-fuse’ capacitor programmed by an oxide-rupture mechanism similar to ReRAM in terms of oxide breakdown however non-reversible [6]. After the initial programming, it works just as ROM [2, 3, 7].
Erasable programmable read only memory (EPROM and EPROM/OTP) was the preceding technology to FLASH memory using the hot-carrier injection physical mechanism. Its added feature over PROM was erasability; however, not electrically. To erase the stored data an ultraviolet light was directed onto the die via the quartz window on top of the chip packaging (see Fig. 1-4) causing ionization within the silicon oxide which allowed the stored charge (in the floating gate) to dissipate [2, 3, 7]. Selective erasure was not possible, and erasure (in general) with UV light was also very unpractical, usually requiring chip removal and taking several minutes; sunlight would erase a chip in weeks, and indoor fluorescent lighting over several years [2, 3, 5, 7]. In light of these challenges and the fact that the quartz window was expensive, manufactures were inspired to create OTP versions of the EPROM with opaque packaging as a means to reduce cost.

Electrically Erasable Programmable Read-Only Memory (EEPROM) has now mostly displaced UV-erased memory by providing 2 main upgraded features: (1) an in-circuit electrically-erasable feature; and (2) the ability to reliably store code and configuration parameters e.g. computer BIOS settings. The EEPROM mechanism is different from EPROM in that it uses a thinner gate oxide layer which enables the chip to electrically erase its own bits, with the help of additional charging circuitry [7].

Most NOR Flash memory is a hybrid style between EPROM and EEPROM mechanisms—programming is through hot carrier injection and erasure is through Fowler-Nordheim tunneling [2, 3]. NOR-based Flash has long erase and write times, but provides access to every single x-y-z address and data buses which allows for random access to any memory location (labeled as the bulk-erase feature in Fig. 1-4). This makes NOR-Flash suitable to replace older ROM technology (like EEPROM) which is mostly code and parameter storage that rarely needs re-programming.
**NAND Flash** memory brings the added feature of faster write/erase speeds for data storage. This fast speed comes at the expense of not having the ability to bulk-erase at the byte-level (compared to NOR Flash). But, on the other hand, it requires less chip area per cell. This renders it very suitable for usage in mass-storage devices such as hard-disk-drive replacements (in the form of solid-state-drives: SSDs).

There are trade-offs in every one of the aforementioned NVM technologies and the ‘missing-link’ is a single-type of high performance memory that combines the fast speeds of NAND Flash, the random-access of NOR Flash, and the massive archival capabilities ($\gg 1$TB) and low cost of modern-day mechanical magnetic storage (15-30 years). Such a device would require a solid-state nonvolatile memory technology that could be manufactured at very high areal densities using some combination of sub-lithographic patterning techniques, multiple bits per cell, and/or multiple layers of devices. As seen in Fig. 1-4, **ReRAM** is proposed as a potential ‘missing-link’ NVM technology enabling a low-power, highly scalable solution with a higher density than (NAND/NOR) Flash. The added operational features that ReRAM is expected to enable are discussed in detail in the next section. Other contending NVM emerging technologies such as PCM, STT-MRAM and CNT-RAM, are briefly summarized in Appendix 1.

### 1.5 Resistive-Switching Memory: Well Positioned as an Embedded Memory?

Embedded memory can be any type of memory seen in Fig. 1-1 integrated on-chip closer to the central processing unit core thus enabling higher-speed and wider bus-width capability which reduces the need for inter-chip communications [8]. Embedded memory technology is generally larger in size and more complex to design and manufacture since it must be made compatible with the process flow for CMOS-based logic. This is a potential advantage for the utilization of ReRAM in that selected-candidate ReRAM technologies are
highly compatible with CMOS processing [8]. ReRAM is well positioned in the embedded NVM space and has been quoted as having the potential to replace embedded DRAM, SRAM and (NOR/NAND) Flash [4, 9]. In addition, ReRAM may also prove to have high performance FPGA [10, 11], logic-in-memory [4] and harsh-environment-compatibility [12] applications. To understand how ReRAM can enable more-efficient embedded and/or normally integrated nonvolatile-memories it is presented in the next section in terms of: (1) its known features and most desirable materials; (2) required specifications for specific types of memory replacement; and (3) how this dissertation was approached and aligned within these terms.

1.5.1 Known Features of ReRAM

Compared to normally-integrated Flash, DRAM and SRAM, resistive-switching memory is unique in terms of it having:

- Only 2 terminals (enabling small die size and very high capacity chips)
- Low mask number requirements for fabrication
- Low temperature, BEOL process flow
- Higher reliability
- Faster speed, lower power
- Byte erasability (as opposed to bulk and block erasability)

As with PCM and most-types of STT-MRAM, ReRAM is a 2 terminal device. However, typical modern-day wafer-level ReRAM testing is done with a 3 terminal one-transistor-one-resistor (1T1R) configuration as seen in Fig. 1-5a. It has been predicted that by using a tunneling bi-layer structure the ReRAM bit-cell I-V curve can be made non-linear [13] which removes the need for the (1T) selector. In recent literature, 2-terminal ReRAM arrays have been demonstrated without the need for a selector device [9, 14].
advantageous since 2-terminal crossbar devices enable the smallest possible die size and very high capacity (3D stackable) chips (Fig. 1-5b-c). A recent publication (from SanDisk/Toshiba) demonstrated a one-diode-one-resistor (1D-1R) configuration for a 24nm node, 32Gb test chip [15], which shows promise as a large-scale, 2-terminal ReRAM crossbar memory structure (Fig. 1-5d).

ReRAM patterning also only requires 1 to 3 masking steps in terms of a sub ReRAM contact-via connection to the ReRAM bottom-electrode, ReRAM stack etching, and thirdly the superior (upper) ReRAM via which connects the ReRAM top-electrode to the rest of the back end of the line (BEOL). The advantages are noticeable in terms of cost reduction when comparing this to embedded Flash which consists of 8 to 11 masking steps due to the Flash cells’ high-voltage-transistors and the need for charge pumps [4, 9].

Figure 1-5: (a) Typical 1T1R, ReRAM test circuit. (b) Smallest unit size of crossbar array. (c) Planar, 2 mask per layer crossbar. (d) SanDisk/Toshiba 32Gb ReRAM test chip [15].
ReRAM is less invasive in terms of modifying the standard fabrication process flow as it can be thought of at a modified metal-1 connection just after the contact-via (see Appendix 2 for 1T1R detailed process flow used in this work). ReRAM also only requires **low temperature BEOL processing** thus minimally affecting any other front-end-of-the-line (FEOL) devices already on chip. FEOL transistor threshold voltages are not shifted due to the added ReRAM process changes, and analog circuitry does not need to be re-simulated and sized when adding the embedded ReRAM option [9]. This also implies that in the future, ReRAM will also be compatible with finFET (3D transistor) technology. Compared to high density NAND Flash reliability (typically 5K cycles, and 1 year data retention [4]) applications, ReRAM could have an advantage, as long as the cell current can be kept low (~1 µA), but this has yet to be proven. The reliability of large area ReRAM dies (in Giga-bytes range) has not yet been proven, and data retention is still an issue within the leading ReRAM materials-stacks.

In regards to **reading speed**, ReRAM read current is in the sub-µA range and has been shown to be as fast as 10 nanoseconds [16] which is comparable to NOR-Flash and SRAM [17, 18]. As for **writing speed**, ≤ 10 ns switching has been reported [19]. It is worth noting that small pulses may be more difficult to propagate depending on chosen array size. Christophe Chevallier, from RAMBUS Inc., as an invited speaker at the 2013 IEEE International Memory Workshop Conference presented this paraphrased statement, “A verify-step may be needed for small pulses due to cell-cell and cycle-to-cycle variations; but nevertheless a ‘part’ of the ReRAM array could potentially be optimized for 10 ns pulse widths while ‘other parts’ of the same array could be ~ 100 ns with the added verify-step”.

In terms of **energy (power/ReRAM cell)** ReRAM, because of its non-volatility, stores only pico-Joules/1-ReRAM-cell [20, 21]. Filamentary-based ReRAM has fast switching speeds, however, higher current and very slow programming speed compared to NAND-Flash [4]. ReRAM requires higher current (compared to Flash
technology) which induces a voltage drop (in terms of I*R) across the cell, and increases leakage from the (still-required) built-in selector devices. To increase performance, companies like RAMBUS Inc. have created ReRAM array sizes which are much smaller than NAND Flash. This is not news since smaller arrays mean shorter (less resistive) word and bit lines, smaller RC time delays, and providing, overall, more power to each cell [9]. The same type of smaller array is not achievable with NAND-Flash as the required decoding circuitry would take too much die area [4, 9]. **Smaller arrays are possible in 1D-1R ReRAM because the required decoding circuitry can be laid out under the cells.**

ReRAM is also predicted to be byte (or even bit) erasable similar to NOR-Flash which would increase overall cell lifetime due to reduced garbage collection and un-needed read/write producers [4, 9]. This added feature makes ReRAM seem similar to NOR-Flash bulk (random-access) erase, but with the added feature of a faster (NAND comparable) system speed.

In summary, ReRAM can potentially bring the following features to market as an embedded or normally-integrated NVM replacement: lowered mask count which reduces complexity and cost; a low temperature BEOL process flow; higher reliability than advanced NAND-Flash; faster write/erase speeds than NOR-Flash; fast read speeds comparable to NOR-Flash and SRAM; random access (byte) erase; and lower overall power. Figure 1-6 compares NAND-Flash and ReRAM in terms of achievable applications. It notes that, 'The Future of NAND-Flash is NAND-Flash!' for low cost USB keys, cards, and mobile device storage. However, for applications such as SSDs and data-centers, where NAND-Flash has performance/reliability issues, ReRAM (and other emerging NVM technologies) could potentially enter the market and preform reasonably (if not expectantly) well.
Which type of ReRAM is the most attractive in terms of materials? The use of fab friendly materials such as HfO$_2$, TiN, TaO$_x$, or amorphous-SiO$_2$ is a requirement as to ensure a low-barrier to entry into today’s modern fabs/foundries.

ReRAM should not compete on cost alone but bring added features similar to its preceding technologies (seen in Fig. 1-4). Start-up companies in search of the ‘killer application’ might concentrate on developing one feature at a time, e.g. (1) long, high temperature data-retention, (2) fast read speed, (3) very low-power etc…

Figure 1-6: List of memory applications comparing (dark-blue) NAND-Flash dominated low-cost market with other applications (some mobile storage devices, SSDs, and data centers) which require the best possible performance/reliability in place of low-cost fabrication. The future of NAND-Flash is NAND-Flash in the low-cost dominated market. However ReRAM and other emerging NVM technologies could take a significant portion of the other markets with rely on performance as well as cost.
1.5.2 Approaching the Problem in More Detail: Needed Specifications

Where is ReRAM technology today? Figure 1-7 shows the breakdown of how ReRAM technology (to-date) compares to each applications’ best-in-class technology. **Embedded-micro-code-automobile-applications** use mostly PROM (fuses/antifuse tech) or NOR-Flash memory. Specifications needed for embedded cells included good data retention (125 °C, 10 years, and 300 °C for 1 minute [4]), read speeds in the range of 10 ns, low density, and low power. ReRAM could reduce wafer cost with its simpler design requirements, reach higher density (only need > 1 MB), and achieve advanced node (< 22 nm) scalability.

ReRAM as a **RAM-like cache memory** is possible with endurance being perhaps the limiting factor. In comparing Figs. 1-7 and 1-8, ReRAM can only compete in a **NAND-Flash storage** application in terms of better performance and reliability yet the limiting factor (to-date) is that large density ReRAM chips have yet to be proven. ReRAM (or MRAM) could open up a new paradigm for **memory usage within logic functions** in terms of nonvolatile logic-in-memory architecture. MRAM has the advantage of infinite endurance in this case. Yet, ReRAM has the advantage of easier integration and comparably fast write-speeds (see Appendix 1 for more info on MRAM). Read disturb needs to be controlled in this case and also reducing the difficulty with applying memory-logic software control algorithms [4].

In terms of **Field-Programmable-Gate-Arrays (FPGAs)**, the ReRAM cell is much smaller than (the typical memory storage unit) SRAM cell and the FPGA array size reduction will translate into shorter lines, and power savings. FPGA code would stay on-chip which is more secure and similar to memory-logic integration [4], and will eliminate the need for a power-up transfer (as with SRAM)[4]. The needed memory window in terms of the ratio between on/off (1 or 0) in this application should be at least 1000, which has been achieved with some types of ReRAM material stacks depending at high current compliance, but further development is needed and endurance at this high ratio level maybe an issue.
Figure 1-8 shows a summary of the most probable entry markets for each type of emerging NVM technology, separating ReRAM into categories determined by filament and non-filament-based operation. These numbers are subject to change and are only estimations summarized from current literature. More details on the other listed emerging NVM technologies can be found in Appendix 1.

<table>
<thead>
<tr>
<th>Application</th>
<th>Embedded μC code, NOR-like</th>
<th>RAM-like Cached data</th>
<th>NAND-like storage</th>
<th>Logic (D-Flip-Flop replacement)</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best in class tech</td>
<td>NOR-Flash, OTP PROM (fuses)</td>
<td>DRAM, PCM</td>
<td>NAND</td>
<td>MRAM</td>
<td>SRAM</td>
</tr>
<tr>
<td>Read access</td>
<td>&lt;50 ns</td>
<td>&lt;50 ns</td>
<td>25 us latency</td>
<td>ns</td>
<td>ns</td>
</tr>
<tr>
<td>Endurance</td>
<td>10 typ.</td>
<td>10⁶ to 10⁷</td>
<td>10⁴</td>
<td>10¹⁻²</td>
<td>100</td>
</tr>
<tr>
<td>Data retention</td>
<td>10 years at 85°C/125</td>
<td>30 days at 55°C</td>
<td>1 year at 70°C</td>
<td>Seconds to years</td>
<td>10 years at 85°C</td>
</tr>
<tr>
<td>RRAM today?</td>
<td>Difficult for the automotive spec</td>
<td>Stretches endurance, possible</td>
<td>Yield of large density chip unproven, possible</td>
<td>Endurance too low, algorithm hard to implement</td>
<td>Possible</td>
</tr>
</tbody>
</table>

Source: Chevallier, C., ReRAM: What are the applications? 2013, IEEE IMW, ReRAM Tutorial Seminar

Figure 1-7: Potential requirements for selected different memory applications comparing ReRAM technology (To-Date: 2013).
Keeping in mind that, compared to other NVM, NAND-Flash has poor performance and reliability, it is still the ‘king’ of low-cost/high-density (≥ 22nm node). The minimum specifications desired by industry which would allow ReRAM to enter the NVM market require ReRAM to be at-least as good as NAND-Flash and preferably competitive with NOR-Flash. Still, the ultimate goal would be an SRAM replacement (which may or may not happen). ReRAM can achieve most of the desired specifications listed below, however not all at one time as there are a few trade-offs. But, ReRAM can be tuned for specific applications if listed requirements in Fig. 1-7 are achieved. Most applications’ specifications for a single ReRAM-cell included [4]:

- Memory window............ On/Off Ratio > 10×
- Memory window............ On/Off Ratio > 100× (Multi-level-On or Off states in one cell)
- Current level .................. < 1m A (at maximum; lower than 50µA is preferable)
- Switching voltages.........< 2 V
- Cycling endurance.........>100 cycles
- Data Retention............(>30 days @ 55 °C) or (10 years at 150 °C)
- Power .......................<1 pico-Joule

Most applications’ **specifications for the ReRAM Array/Chip/System** include [4]:
- Latency .......................0-50 µs
- Read speed..................10 ns to 50 ns
- Write speed..................< 10 µs/bit
- Endurance .................10 to 10^{16} cycles
- Power ..........................0.1 to 10 pJ/bit

In addition to these required specifications, ReRAM must also show integration, scalability, statistical reliability, relevant circuitry design (e.g. standard libraries), and system solutions (e.g. standard sub-system configurations). **Integration challenges** include, as mentioned in section 1.5.1, a fab-friendly material set, as well as avoiding the use of integration tools that could potentially contaminate the ReRAM stack. **Scalability** must be demonstrated as a means of determining the lower limit at which switching becomes probabilistic (or enhanced) at lower dimensions. **Statistical reliability** must be improved for large yields. **Design challenges** include understanding the importance of current compliance (I_{COMPL}) and determining the most practical method of implementing it. For example, are there trades-offs between lowered I_{COMPL} and higher voltage? Can we take advantage of ReRAM cell variability; how well can we control it? If not, can we understand the underlying physical mechanism(s) well enough to reduce ReRAM variability to a controllable level? **System challenges** include the reduction of testing-time with regard to the initial activation of each cell(s) filament i.e. FORMING.
1.6 Document Organization

In the preliminary stages of this dissertation work, the hafnium-oxide (HfO₂) dielectric was chosen as the fab friendly material (see Chapter 2) and ReRAM cells were fabricated with the help of the SEMATECH Austin-Texas-based clean-room, and the SUNY College of Nanoscale Science and Engineering’s 300mm research fab. In terms of the aforementioned industry-required specifications and other deemed challenges:

**Chapter 2** briefly covers (for completion) the to-date ReRAM taxonomy, preferred switching materials and atomic level mechanisms. Preliminary works are discussed which include the benefits of HfO₂ as a ReRAM switching material, the critical requirement for sub-stoichiometry, and the preliminary electrical, physical and theoretical analysis.

**Chapter 3** focuses on fundamental experimental investigations of ReRAM operations. It covers filament-formation-characterization in terms of defining the effects of current, voltage and temperature on (low and high power) switching stability. A novel ‘hot’ FORMING method is developed as to improve (RESET/SET) switching uniformity which is followed by ac and dc endurance characterization.

**Chapter 4** uses the previous chapters’ experimental results as ‘calibration’ parameters to create a simulation model that encompasses all the necessary components of filament growth which are dependent on current, temperature and voltage. These include ion/vacancy pair generation, oxygen ion diffusion, recombination and the importance of sub-stoichiometric HfO₂. This simulation approach introduces a novel method of including intrinsic variability and explaining ReRAM switching which is dominated by oxygen ion (O⁻²) diffusion.

**Chapter 5** is a summary and discussion of the presented results and suggests future work. **APPENDIX 1** Briefly summarizes the salient functionality of other emerging NVM technologies including PCM, STT-MRAM, and CNT-RAM. **APPENDIX 2** steps through the
processing integration steps for the 1T1R ReRAM testing vehicle used in this work. The ReRAM cell is fabricated as the metal-1 contact above the transistor drain and contact via.

**APPENDIX 3** describes how 3D grains and grain boundaries were created in MATLAB. Code is in Appendix 5. **APPENDIX 4** further discusses the numerical methods mentioned in chapter 4 (and implemented in appendix 5) as to solve a linear system of differential equations. **APPENDIX 5**: Complete Matlab code used in chapter 4 simulations.
CHAPTER 2: Background and Preliminary Work

2.1 ReRAM Taxonomy Background and Preliminary Physical Analysis of Filament

Resistive switching memory (ReRAM) is currently under consideration because of its superior scaling opportunities [22-24] and its potential to replace modern day NVM technologies (see Chapter 1). ReRAM offers the possibility of reversible changes in the conductivity of a dielectric film between high (HRS) and low (LRS) resistance states by successively applying voltage pulses of either different (bi-polar) or similar polarities to the metal-insulator-metal (MIM) memory cell [20].

2.1.1 Background: Brief History

Figure 2-1a displays a snap-shot of the very first resistive switching papers published in the 1960’s where the authors linked their discovery with a potential NVM application. Figure 2-1b shows key historical dates which followed ‘33 years later!’ ending with D.B. Strukov’s famous 2008 Nature publication which sparked the current interest in ReRAM technology.

![Figure 2-1: Historical recount of resistive memory discovery.](#)
2.1.2 Background: RRAM Switching Materials: Transition-metal-oxides (TMOs)

Today, there has been an explosive effort focused on resistive-memory research. As can be seen in Fig. 2-2, ‘everything-more-or-less-switches’ and the categorization of all ReRAM switching materials (and electrodes) can be a daunting task. Figure 2-2b displays a pie chart showing a ‘handful’ of transmission metal oxides (TMOs) that are the main stream focus of ReRAM development (TiO$_x$, HfO$_x$, NiO$_x$, TaO$_x$) due to their high potential to become a reliable NVM replacement. Note these are all sub-stoichiometric in nature (discussed further in section 2.3). Figure 2-2c highlights the periodic table in terms of published switchable dielectric materials and also a list of published electrode materials.

![Image of RRAM Switching Material Categories and Pie Chart](image)

Table Ref: Deepak C. Sekar, RRAM Technology and Market Opportunities and IMW 2013; Pie Chart (Google Scholar 2012, Jan-2012, Dec)

![Image of Periodic Table](image)

Figure 2-2: (a-b) ReRAM switching materials and (c) ReRAM switching dielectrics and electrodes materials.
2.1.3 Background: Atomic-Level Mechanisms

Although electrically induced in all cases, the actual physical driving forces of resistance-switching memory can be quite different depending on the materials and device structure. In terms of atomic-level mechanism taxonomy, Fig. 2-3 categorizes 5 different types of physical mechanisms of this type of memory: Phase Change; Thermochemical; Valence; Electrochemical; and Electrostatic/Electronic.

These mechanisms are shown to be dominant in different materials and produce unipolar (similar polarities) or bipolar switching. The primary mechanism for phase change materials (PCMs) are temperature induced changes between a crystalline phase (ON-state) and an amorphous phase (OFF-state) (PCM-summary-Figure in Appendix 1). PCMs are primarily unipolar in switching nature. ECM stands for electrochemical...
metallization (and is commonly referred to as conductive bridge or CB-RAM). It relies on an electrochemically active electrode metal such as silver (Ag), and the drift of the highly-mobile Ag cations through the ionically conducting dielectric layer. This leads to Ag dendrite growth on the counter electrode which forms a highly conductive filament between both electrodes (creating the ON state or low-resistance state). Upon applying an opposite polarity voltage, an electrochemical dissolution of these filaments takes place, [23, 24]. The redox processes in the metal-insulator-metal (MIM) cell is either triggered by temperature, voltage or both [23, 24]. The valence change mechanism (VCM) occurs mostly in specific transition-metal-oxides (TMOs) and is triggered by a migration of anions, such as oxygen anions [23, 24]. A subsequent change of the local stoichiometry leads to a redox reaction expressed by a valence change of the cation sub-lattice and a change in the electronic conductivity [23, 24].

VCM-TMO-based-ReRAMs are typically bipolar in nature. The thermochemical mechanism (TCM) leads to a change of the stoichiometry due to a current-induced increase of the temperature. Electrostatic/electronic mechanisms (EEM) are dominated by static charge or electron flow.

Some materials including transition-metal-oxides exhibit a mixture of the aforementioned mechanisms. And the difference between TCMs VCMs and PCMs may be perplexing when one starts trying to differentiate between phase changes, and (sub-)stoichiometric properties. The PCM effect is closely related to the three redox-based effects (TCM, VCM, ECM) [24]; and thus, it is better to comprise PCM, TCM, and VCM systems into one group in which transition-metal-oxides (TMOs) and chalcogenides both dominate. (Chalcogenides are mostly used for PCM based memory and consist of, by definition, chemical compounds having at least one element from group 16 (old-style: VIB or VIA) of the periodic table.) These chalcogenides materials accommodate up to 25% per-volume of
vacancies in the crystalline (ON-state) phase [24]. Yet, during switching, the kinetics of the phase transformation is accelerated by several orders of magnitude due to the presence of these defects [24]. Lattice defects play a similarly essential role in the redox-related memory effects as seen by Hewlett Packard’s early work with (TMO) TiO₂. As seen in Fig. 2-4a electro-FORMING created vacancies by ‘ripping’ oxygen away from the TiO₂ lattice and inducing the evolution of O₂ gas bubbles inside the material [25]. Small permanent deformations remained after (this type of high voltage) electro-FORMING with a physical mechanism of TiO₂ changing phase into Ti₄O₇ and remaining stable at room temperature (see Fig 2-4b). Hewlett Packard observed this (Ti₄O₇) ‘Magneli phase’ via TEM analysis [26] also seen in Fig. 2-4c. This suggests that TiO₂ is driven by VCM, PCM and TCM forces. Therefore to clarify this taxonomy it is better to distinguish categories of materials (such as TMOs or any other material category listed in Fig. 2-2a) which may have a number of different contributing atomic mechanisms. Binary TMOs (as seen in Fig. 2-2a as the leading types of ReRAM materials under research) can either primary mechanisms of ionic redox-related chemical reactions, thermal or electronic effects depending on the electrode dependence of the dielectric materials [27, 28], thus inducing single or multiple conductive-filament types of geometry (Fig. 2-3) which may or may not be area dependent: see next section.

2.1.4 Background: Distinguishing Filamentary Switching and Area Distributed Switching

This section clarifies the two fundamentally different geometrical localizations of the switching event: the filamentary-switching scenario and the area-distributed switching (non-filament) scenario. In filamentary switching the ON resistance is independent of the electrode area due to the filaments very small size. In the non-filament scenario, the change in resistance occurs more or less homogeneously over the entire area of the memory cell electrode, hence leading to an ON resistance which is proportional to the electrode area.
Filamentary switching is preferred by industry as it will enable extreme scaling of the memory device; and thus it is the focus of this work (and most other ReRAM works to date). In the literature, recent publications (see Fig. 2.5) have been presented giving direct evidence of a single filament as being the dominating switching mechanism in filamentary TMO-based ReRAM [29].

Figure 2-4: (a) Electro-FORMING of vacancies via ‘rippling’ oxygen from the TiO$_2$ lattice and induction of O$_2$ gas bubbles inside the material. Small permanent deformations remained after (this type of high voltage) electro-FORMING with (b-c) physical mechanism of TiO$_2$ changing phase into Ti$_4$O$_7$ and remaining stable at room temperature.
Figure 2-5: Direct evidence published at IEEE IMW 2012 of one dominant filament as the active mechanism in TMO based RRAM. Analysis was done with transmission electron microscopy and electron energy loss spectroscopy.
2.1.5 Preliminary Physical Analysis: Spectroscopy Used to Define Filament Cross-Sections

In this study, we have employed scanning transmission electron microscopy (STEM) and electron energy loss spectroscopy (EELS) to observe electrical-stress-formed conductive-filaments in HfO$_2$-based crossbar devices with a Hf top-metal gettering i.e. oxygen exchange layer (OEL) onttop of the HfO$_2$ dielectric, and TiN electrodes. Scanning-transmission-electron-microscope (STEM) and electron-energy-loss-spectroscopy (EELS) were both used to verify the dominating filament-based mechanism in W/Hf/HfO$_2$/TiN 50×50 nm$^2$ HfO$_2$-based ReRAM devices. To study the morphology and microstructure of a conductive filament formed under these conditions a 50×50 nm$^2$ crossbar device was analyzed by TEM. In order to include the entire device volume a specimen (~60 nm thick) was prepared by using a focused ion beam (FIB) tool (in collaboration with S. Privitera and S. Lombardo in IMM, Catania, Italy; this work was published in 2013 [30]).

Figure 2-6 shows a large view of the filament which was seen with STEM and EELS. Figure 2-7 shows a more detailed (summary) analysis of the same filament (Fig. 2-7b). Figure 2-7a shows the preliminary I-V sweep carried out at room temperature, during which the formation of the conductive filament occurred (~1.4 V). After filament formation, the device exhibited much higher conductance which linearly decreases with temperature indicating metallic (LRS) characteristics (plotted in Fig. 2-8a). Figure 2-7b and c show two different device examples of STEM dark field micrographs indicating a bright ‘semiconically’ shaped region in the HfO$_2$ oxide. The contrast in these images is determined by the local electronic density, i.e. the local atomic number (Z) per unit-volume. Consequently, a brighter area in this image indicates a higher average Z region in the HfO$_2$-dielectric layer meaning a metallic Hf-rich filament in HfO$_2$, as predicted by the plot in Fig. 2-8a and [20].

This STEM image was then combined with EELS data and collected point-by-point (< 1 nm diameter electron beam spot with pixels of 1 nm×1 nm) throughout the image area.
It should be noted that electron energy loss at low energy is mainly dominated by plasmon losses and sensitive to the local chemical composition. (Plasmons are quasiparticles which result from plasma oscillations just as photons and phonons are quantizations of electromagnetic and mechanical vibrations [31]). Figure 2-8b shows electron energy loss spectra collected in exemplary regions containing Hf, or HfO₂ respectively. Each spectrum can be fitted as a sum of Gaussian functions having the peak-position, width and height characteristic of their respective materials [30]. Due to a higher atomic scattering factor associated with higher atomic numbers, such as Hf in this case, the intensity of a signal acquired in the regions containing pure metallic Hf is correspondingly much lower [30].

A 3D EELS image was also obtained by collecting point by point spectra throughout the image area [30]. Such an image was analyzed as the superimposition of the plasmon losses of Hf, and HfO₂ each with a specific ‘weight’, which plays the role of a fitting parameter [30]. Figure 2-7(d and e) shows the contour EELS map determined from fitting the 3D spectral image acquired from the (STEM imaged region) circled in Fig. 2-6. In Fig. 2-7d a lack of the HfO₂ signal is observed around the central region of the device dielectric, with a corresponding increase of the metallic Hf signal seen in Fig. 2-7e (indicated in Fig. with black arrows). The Hf metal counts plotted in an intensity profile are shown in Fig. 2-6 and in Fig. 2-7(f and g). Figure 2-7(f and g) correspond to the red-dotted and yellow-dotted lines in Fig. 2-7b. In comparing both intensity plots there is an indication of a conical shaped filament with estimates of top and bottom filament cross sections of 5.5 nm and 2.5 nm, respectively. These results appear to be in good agreement with the creation of a dominate Hf metal rich filament which has varying sized cross-sectional areas.
Mechanism: HfO$_2$-based RRAM filament “seen” with STEM / EELS

- Filament is:
  - Conical
  - 2.5nm at bottom
  - One dominant CF

- RRAM conductive filament

- Hf-O breaks along grain boundary; O diffusion

- Understanding needed for scalability, variability, endurance

Figure 2-6: Mechanism for HfO$_2$-based ReRAM is dominated by one filament.

Figure 2-7: Transmission-electron-microscope (TEM) and electron-energy-loss-spectroscopy (EELS) as a means to verify the filament-based mechanism of HfO$_2$-based ReRAM

Conical shape as predicted based on defect generation model; higher power dissipation closer to anode: G.B. et al., IEDM 2010
2.1.6 Preliminary Definitions: Defining TMO-ReRAM in terms of Filament Formation

In a majority of the transition-metal-oxide based ReRAM stacks the resistive switching is controlled by the formation and dissolution of a conductive filament in the dielectric [21, 24, 32-34]. This section aims to give a primary overview of how to understand the I-V characteristics for bi-polar switching which will be heavily referenced throughout the remaining portions of this dissertation. Figure 2-9 shows the typical current (Fig. 2-9a) and semi-log current (Fig. 2-9b) scales used. The FORMING step seen in red in Fig. 2-9b is different from the electro-FORMING event previously described and will be further discussed in terms of mechanisms in section 2.5. Key FORMING related definitions included FORMING voltage ($V_F$) and FORMING current ($I_F$). The 1st RESET after FORMING is also in red (in Fig.2-9b) and useful in characterizing the previous FORMING event.

Figure 2-9a illustrates the conductive filament after FORMING between the top and bottom electrodes. This filament in the case of transition-metal-oxides is illustrated here as being made up of oxygen vacancies ($V^+/V^{+2}$) which is in agreement with the previously
discussed preliminary physical analysis. The low-resistive-state (LRS) and the high-resistive-state (HRS) as depicted in Fig. 2-9a are measured at a 0.1 V read voltage along their respective RESET and SET curves. Figure 2-10 steps through FORMING/RESET and SET in more detail. Figure 2-10(1) describes the filament formation starting with the device in a fresh-state with no applied voltage stress and an initial leakage current which depends on the dielectric stoichiometry. FORMING the device requires the application of an applied voltage (positive in this case), inducing current and dielectric breakdown thus changing the device from HRS to LRS, i.e. FORMING the conductive filament (CF) between the two electrodes. In Fig. 2-10(2) the opposite voltage polarity is applied, and for a short amount of time the device is still in LRS with the CF still connecting the top and bottom electrode. In Fig. 2-10(3) the device has switched from LRS to HRS meaning the applied voltage has induced a ‘break’ in the connection between the CF and the electrodes. In Fig. 2-10(4) a positive voltage sweep is re-applied, and again for a short amount of time the device is still in HRS. In Fig. 2-10(5) the CF has been re-connected between the top and bottom electrode completing 1 full cycling event. The details of the voltage/current and temperature dependences of these described events are further discussed below.

![Figure 2-9: Basic definitions and IV curves for the FORMING event in filament-based TMO-ReRAM.](image)
Figure 2-10: Filament formation starting with 1) the device in a fresh-state with no applied voltage stress and an initial leakage current. FORMING the device requires the application of an applied voltage, inducing current and dielectric breakdown and thus changing the device from HRS to LRS, FORMING the conductive filament. 2) The opposite voltage polarity is applied, the device is still in LRS and the conductive filament (CF) is still connecting the top and bottom electrodes. 3) Device is in HRS. The CF has broken. 4) A positive voltage sweep is applied and the device is still in HRS. 5) The CF is re-connected between the top and bottom electrode completing 1 full cycling event.

2.2 Background: Defining the Benefits of Hafnium-Oxide as a RRAM (TMO) Dielectric

There are many choices of transition-metal-oxide (TMO) based ReRAM devices with HfO$_x$, TaO$_x$, and WO$_x$ in the forefront of the race for an industry standard ReRAM material [35]. HfO$_x$-based ReRAM using a thin Ti oxygen gettering layer called the oxygen exchange layer or OEL has shown strong potential for the next generation nonvolatile memory as it doesn’t require the use of noble (precious) metals for electrodes like other TMO based ReRAM devices. It has low leakage currents (which are beneficial for determining the difference between ON (LRS) and OFF (HRS) current), it is fab-friendly, and in terms of performance it also has stable nanoscale switching capabilities (Fig. 2.11). In subsequent
chapters, the ReRAM stack that will be analyzed and discussed will have the following material layers: TiN/OEL/HfO$_2$/TiN.

![HfO$_2$ based-ReRAM has many benefits](image)

**Figure 2-11**: Parameters detailing the potential benefits of hafnium oxide based resistive memory.

2.3 Preliminary Work: Critical Requirement of Sub-Stoichiometric (HfO$_x$) Dielectric

As was mentioned in section 2.1.2 the role of defects is critical in ReRAM switchable TMOs, specifically HfO$_x$ as inferred by the sub-stoichiometric sub-script. The *FORMING* curve and subsequent cycling characteristics for a “perfectly” stoichiometric hafnium-oxide dielectric and for a “defective” sub-stoichiometric HfO$_x$ are displayed in Fig. 2-12 for comparison. This figure shows that well-behaved ReRAM switching is only observed for sub-stoichiometric dielectrics (Fig. 2-12a). A possible explanation for this difference in ability to reproducibly change resistance is schematically illustrated in Fig. 2-13. When voltage is applied to the "leaky" sub-stoichiometric HfO$_x$, filament formation is more gradual due to a smaller voltage drop across the dielectric leading to smaller overshoot current (see next section for more details about controlling overshoot-current). For stoichiometric HfO$_2$ the *FORMING* voltage is higher, resulting in a higher overshoot current with greater energy dissipation (~I•V) during the dielectric breakdown when the filament is formed and, subsequently, a larger filament cross-section that is more difficult to *RESET*.
to a higher resistance state. This is only one possible explanation which doesn’t fully take into account the microscopic details of the dielectric material as the defects inside the dielectric could also play other roles. This is a focus of subsequent chapters. The main point is that sub-stoichiometric HfO₂ switches much more readily than stoichiometric HfO₂ which suggests a defect assisted, ionic-movement based mechanism.

Figure 2-12: Representative FORMING and cycling for a reduced sub-stoichiometric HfOₓ dielectric (a) and a near perfect HfO₂ dielectric (b). The near perfect stoichiometric HfO₂ has difficulty to RESET, but the more defective HfOₓ demonstrates well behaved ReRAM switching.

Figure 2-13: Cartoon analogy using leaky or non-leaky buckets to understand FORMING differences in leaky-HfOₓ (a) and stoichiometric HfO₂ (b), respectively. The lower cartoons represent current vs. voltage traces from oscilloscope monitoring showing overshoot differences at FORMING. The stoichiometric HfO₂ has higher FORMING voltage and current overshoot.
2.4 Preliminary (HfO$_x$) Electrical Analysis

2.4.1 One-Transistor, One-Resistor (1T1R) Testing Structure: Limiting Overshoot-Current and Parasitic Capacitances

Current-overshoot, or the maximum current during the initial filament formation event, needs to be controlled to prevent an irreversible thermodynamic dielectric breakdown (caused by a run-away current) [32]. In practice, a maximum current level is often enforced by setting a current compliance limit (I$_{COMPL}$) in the parametric analyzer controlling the device under test. However, the analyzer’s response-time, which is both software and hardware dependent, is usually longer than the characteristic FORMING-event-time and leads to a current-overshoot which is higher than the desired current-compliance limit. This can be driven by parasitic capacitance. As a consequence, the overshoot results in a higher required RESET current [36], which, in turn, increases power consumption and limits HRS-current (see Chapter 3). As mentioned in Chapter 1 for actual applications, the operational current is controlled by a transistor or diode in series with the ReRAM device and is expected to be much more effective against overshoot. Figure 2-14 describes in detail the effects of overshoot-current. The circuit diagram, in Fig. 2-14a shows the parasitic capacitances in the externally-connected-1T-1R ReRAM testing-setup. The externally applied transistor still adds unwanted parasitics which affect the I$_{max}$ 1st RESET and, below a 100 $\mu$A current-compliance, other steps can be applied (besides integrating the 1T1R testing-setup) in terms of reducing I$_{CP}$ (see Chapter 3). Figure 2-15 indicates the overshoot current for 50 $\mu$A compliance during FORMING operation, as measured by an oscilloscope, is near 480 $\mu$A for this particular HfO$_x$ film, with a FORMING voltage near 2 V. Figure 2-14b shows a red-circle over the I$_{max}$ 1st RESET current. this value is used to gauge the total amount of overshoot current that occurred during the FORMING event as, typically,
the parameter analyzer is not fast enough to stop or display all the details of a given FORMING-IV curve. Figure 2-14c plots data comparing the impact upon \( I_{\text{max}} \) 1st \( \text{RESET} \) from an extreme \( \text{FORMING} \) event. Note that the HRS for both the 1R and external 1T1R cases are both limited (higher in value) compared to the integrated 1T1R case. The integrated-1T1R case (1T1Rint) shows an \( I_{\text{max}} \) 1st \( \text{RESET} \) lower than the applied \( \text{FORMING} \) compliance. The reason for this is complicated and requires simulation results. It will be discussed further in Chapter 4. Figure 2-14d compares data collected for this work with the referenced seminal paper [36] data. This plot compares the 1R and 1T1R testing setups and their respective ability to reduce the current overshoot during \( \text{FORMING} \). The 1T1R testing setup clearly allows for the shown linear trend between \( I_{\text{max}} \) 1st \( \text{RESET} \) (y-axis) and the \( \text{FORMING} \) current compliance limit (x-axis).
Figure 2-14 (a) Circuit diagram of parasitic capacitances in the 1T1R -ReRAM testing-setup. $I_{CP}$ is the total capacitive charging current and is a function of the product ($dV/dt \times$total circuit capacitance). ReRAM current is the sum of current through the transistor and the capacitor charging current. This figure is referenced from Ref. [37]. (b) Shows a red-circle over the $I_{max \ 1^{st} \ RESET}$ current. This value is used to gauge the total amount of overshoot current which happened during the FORMING event. (c) Shows real data of the impact of $I_{max \ 1^{st} \ RESET}$ from extreme overshoot during the FORMING event. (d) Shows data from this dissertation in reference to the seminal paper [36]. This plot compares the 1R and 1T1R testing setups and their respective ability to reduce the current overshoot during FORMING. The 1T1R testing setup clearly allows for the shown linear trend between $I_{max \ 1^{st} \ RESET}$ (y-axis) and the FORMING current compliance limit (x-axis) [32].
Figure 2-15: Real-time monitoring of the current-overshot. This device was formed at ~50 µA and tested in the 1R (no transistor setup) as to maximize the overshoot effects. The overshoot current captured by the oscilloscope is ~ 480 µA which is 8× the desired current compliance.

Examples of well-behaved cycling characteristics – which are achievable for a HfOₓ-based ReRAM when current-overshoot is mitigated – are displayed in Fig. 2-16. This data was collected with the externally connected 1T1R testing setup seen in Fig.2-14a and the observed maximum reset current is only marginally above the compliance limit of 1 mA and slightly above the 50 µA I_{COMPL} current level applied during the FORMING/SET operations.
Figure 2-16: Representative FORMING and cycling for a reduced sub-stoichiometric HfOx dielectric (A), or a near perfect HfO2 dielectric (B). The near perfect stoichiometric HfO2 has difficulty to RESET, but the more defective HfOx demonstrates well behaved ReRAM switching.

Figure 2-17 shows various HfOx-based (labeled MeOx) ReRAM devices that were tested in order to further understand the relationships between FORMING voltage, current-overshoot and reliable switching. HfOx (=MeOx) was labeled this way as not to confuse the Ti OEL layer (not shown here) which was used to make HfO2 sub stoichiometric, with the added Al2O3 layers. When thin Al2O3 layers were included in the stack next to the dielectric layer, RESET was not observed for any of the tried ICOMP limits: 10 µA, 100 µA or 1 mA. This result can be further understood by examining the overshoot currents for each stack (shown in Fig. 2-18). Placing a lower dielectric-constant material (Al2O3) next to the HfOx layer increased the FORMING voltage and the overshoot current. This suggests that the FORMING voltage and overshoot-reduction both play an important role in defining the filament properties and the amount of the energy released during FORMING [32].
Figure 2-17: Various ReRAM stack configurations investigated for the effects on FORMING and overshoot.

Figure 2-18: Trend of FORMING voltage vs. overshoot current as measure by oscilloscope for HfO<sub>x</sub> compared to the same HfO<sub>x</sub> having various thin Al<sub>2</sub>O<sub>3</sub> layers added. The higher FORMING voltage and current overshoot for the Al<sub>2</sub>O<sub>3</sub>|HfO<sub>x</sub> stacks resulted in an inability for resistive switching.

Figure 2-19 further investigates the dependence of FORMING-voltage (V<sub>F</sub>) and overshoot-current on ReRAM stack configurations by comparing uncontrolled and mitigated overshoot. All current compliance levels in Figs. 2-19(a, and b) were applied only
with the parameter analyzer (no 1T1R configuration.) Devices investigated for the data in Fig. 2-19 (a, b, and c) had a current compliance limit of 50 µA. Unfortunately, there was limited data extracted from this preliminary study. However, Fig. 2-19a shows all stacks as having a clear reduction of current-overshoot with $V_F$. ‘Good switching’ can also be further interrupted as more ‘defective’ sub-stoichiometric films having a lower amount of energy released during FORMING (due to a lower FORMING voltage as theorized in Fig. 2-13) and lower current-overshoot.

Figure 2-19b shows the same data re-plotted for all device areas. In comparing this figure with Fig. 2-19a I can be understood that higher $V_F$ leads to higher current overshoot but that overshoot also depends on ReRAM stack configurations (in comparing ‘good’ vs ‘poor’ switching).

In Fig. 2-19c more devices were measured from the same device-stack-types and compared in terms of un-controlled current-overshoot and 1T1R mitigated current overshoot. This figure shows that the use of the external 1T1R reduced the overshoot, however, it did not increase the performance of the subsequent device switching. This further indicates that the ReRAM dielectric properties, in terms of the level of sub stoichiometry and stack layer configuration, both play key roles in enabling repeatable ON/OFF switching.
Figure 2.19(a) Dependence of $V_F$ and current overshoot ‘density’ vs. different ReRAM stack configurations and device sizes. (b) Dependence of current overshoot on $V_{FORMING}$. (c) Current overshoot is mitigated with 1T1R.
Figure 2-20: FORMING I-V curves for various thickness but same sub-stoichiometry of HfO_x. Thicker HfO_x films have lower fresh state leakage and higher FORMING voltage as expected. FORMING and RESET/SET cycling for various thicknesses HfO_x-based ReRAM devices with increasing thickness from A to D. The % overshoot in RESET current above compliance from oscilloscope monitoring is listed for each. Very thin films have high fresh state leakage, no clear FORMING, and do not RESET. Thicker films display FORMING and show well behaved cycling. Thickest films have lowest fresh state leakage and higher FORMING voltage and do not show ability to RESET.
2.4.2 Scaling Trends with Material Layer Thickness and Device Area

The effect of HfO$_x$ thickness on $V_F$ is shown in Fig. 2-20a for films ranging in thickness from ~ 60 Å to ~ 160 Å. As expected, the leakage in the fresh device decreases, and the $V_F$ increases with increasing HfO$_x$ film thickness. Figure 2-17b shows the typical RESET/SET cycling results for these various HfO$_x$ films. For the thinnest HfO$_x$ films, the fresh state leakage was very high, and neither FORMED nor RESET. For the slightly thicker HfO$_x$ films (labeled (b) in Figs. 2-20a, and b), despite the fresh state leakage being relatively high, these device-types RESET to a higher resistance state and reproducible cycling is observed. As the HfO$_x$ becomes even thicker, FORMING events become more obvious, and RESET with reproducible cycling is observed. Eventually, for the thickest HfO$_x$ film with the lowest initial leakage and the highest $V_F$ (labeled d in Fig. 2-20b), no RESET is observed due to high overshoot-current during the FORMING event (referring back to Fig. 2-19).

HfO$_x$ thicknesses in these examples reduce as the Ti OEL layer increases (notice the illustrated ‘gap’ in Fig. 2-17a and b cartoons) which implies a varying level of sub-stoichiometry inside the switching dielectric. These results will be correlated with simulations in Chapter 4 Section 4.4.3.

As a means to more clearly display the trends between $V_F$, device size, and (Ti) OEL induced sub-stoichiometry, Fig. 2-21 summarizes all these effects with the HfO$_x$ thickness held constant. Figures 2-21a, b and c compare different levels of induced sub-stoichiometry. Figure 2-21a shows both an increase in $V_F$ variation, along with an overall decrease in $V_F$ magnitude as OEL thickness increases. Figure 2-21b relates the fresh state leakage current with $V_F$ for every different type of ReRAM stack (illustrated in Fig. 2-21). Figure 2-21b’ plots 4 different-device-sizes (from the same lot) with ReRAM-stack-type held constant. Comparing figs. 2-21(b and b’) indicates that as the device size (in b’) increases so do the number of potential leakage-current paths between the top and bottom electrodes. These
two figures also indicate that this increased leakage current with device size is similar to increased leakage current with changing the HfO$_x$/Ti-OEL thickness ratio i.e. level of sub-stoichiometry.

Figure 2-21b” takes this analysis one step further by relating the subsequent memory window (HRS/LRS ratio read at 0.1 V) from the exact same devices plotted in Fig.2-21b’ with forming voltage. *Figure 2-21b” shows that not only does $V_F$ increase with a reduction in device size but that the overall memory window also reduces respectively with device size.* This trend was further confirmed on a different ReRAM-stack-type (Zr OEL) having a different HfO$_x$/OEL thickness ratio; results are seen in Figs. 2-22 through 2-25. Figure 2-25 plots the summary of Figs. 2-22 through 2-24 in terms of a Gaussian distribution for each of the device-sizes’ respective memory window ratios.

*Noting that in Fig. 2-21b’ $V_F$ increases with reduced fresh state leakage current.* Figure 2-26 plots the exact same devices seen in Figs. 2-21b’ and b” overlaid against more devices of the same ReRAM-stack-type but with an increased OEL thickness (yellow data points). HfO$_x$ thickness was constant. This increase in OEL thickness for the same HfO$_x$ thickness increased the amount of defects present in the HfO$_x$ film. This increased the fresh state leakage and reduced the $V_F$ of the 50×50 nm$^2$-sized devices (compared to the plotted red data points of the same 50×50 nm$^2$ size). This figure also indicates that the subsequent memory window increased for the 50×50 nm$^2$ sized devices. *This highlights the significance of the induced defects inside the dielectric film which again points toward a vacancy driven mechanism for ReRAM switching in these device configurations.*

This preliminary electrical work raises a host of interesting questions related to the physical and electrical characteristics of these ReRAM devices. These results will be further analyzed experimentally in Chapter 3 and through simulations in Chapter 4. But, before
presenting those details it is important to review the dielectric breakdown theory which has been proposed for ReRAM-switching.

Figure 2-21: Summary of scaling trends. (a, b and c) show trends for different HfO$_x$ /OEL thickness ratios with HfO$_x$ thickness held constant. (b’ and b”) expand on (b) by comparing $V_F$ and memory window trends increasing device size for a constant HfO$_2$/OEL ratio. Device sizes in B’ and B” range from 50×50 nm$^2$ to 2000×2000 nm$^2$. Memory window (HRS/LRS ratio) taken at 0.1 V read.
Figure 2-22: All three plots group different device sizes, respectfully. Each box plot shows 5 different devices in terms of ‘LRS/HRS ratios’ i.e. memory windows. Device size is constant for each plot and ReRAM stacks are identical. The x axis represents 5 different devices and also indicates the cycle in which the device stopped switching. $I_{compl}$ was ~85µA for each device and induced with an external 1T1R setup. Each LRS and HRS point was read at 0.1V. Each point next to each box plot is the LRS/HRS ratio for ONE RESET/SET cycle. ReRAM stack (TiN/Zr/HfO2/TiN).
Figure 2-23: All three plots group different device sizes, respectfully. Each box plot shows 5 different devices in terms of 'LRS/HRS ratios' i.e. memory windows. Device size is constant for each plot and ReRAM stacks are identical. The x axis represents 5 different devices and also indicates the cycle in which the device stopped switching. I_COMPL was ~85µA for each device and induced with an external 1T1R setup. Each LRS and HRS point was read at 0.1V. Each point next to each box plot is the LRS/HRS ratio for ONE RESET/SET cycle. ReRAM stack (TiN/Zr/HfO2/TiN).
Figure 2-24: One more device size. Each box plot shows 5 different devices in terms of 'LRS/HRS ratios' i.e. memory windows. Device size is constant for each plot and ReRAM stacks are identical. The x axis represents 5 different devices and also indicates the cycle in which the device stopped switching. I\textsubscript{COMPL} was ~85µA for each device and induced with an external 1T1R setup. Each LRS and HRS point was read at 0.1V. Each point next to each box plot is the LRS/HRS ratio for ONE RESET/SET cycle. ReRAM stack (TiN/Zr/HfO\textsubscript{2}/TiN).

50 Precentile (Center of Gaussian curve)

Figure 2-25: Summary plot of Figures 2-22 through 2-25. This figure plots the 50th percentile ‘center’ of a Gaussian distribution for the ‘LRS/HRS ratios’ (memory windows) seen in Fig. 2-22 through 2-25. Each point in this plot is ONE device’s median memory window. The x axis represents each respective device size.
2.5 Preliminary Theoretical Analysis

2.5.1 Introduction to the Dielectric-Breakdown-based RRAM Model

This section summarizes the seminal dielectric-breakdown, and filament-based metal-oxide resistive memory papers [20, 21, 23, 24, 33, 38-54]. These papers combine electrical, physical, and transport/atomistic modeling and experimental results and represent the fundamental theory behind this work. They identify critical conductive
filament mechanisms governing HfO$_2$ dielectric films and links them with resistive switching memory.

2.5.2 Theoretical FORMING/RESET/SET Switching Process Overview

To describe the material changes in a dielectric associated with resistive switching, one must start with the conductive filament (CF) formation which ultimately determines the CF properties [20]. The CF FORMING process in HfO$_2$ is a manifestation of dielectric breakdown (BD), an abrupt formation of a localized region between the electrodes within which the dielectric composition becomes more oxygen-deficient [53] rendering the region conductive (the CF formation) [20]. Dielectric breakdown represents a “weak link” event, controlled by the first conductive path formed between the electrodes. Therefore, it is expected to follow Weibull statistics [53]. The FORMING process is found to define the filament geometry, which in turn determines the temperature profile and, consequently, the switching characteristics [21]. The FORMING process is analyzed within the concept of dielectric breakdown [53] which exhibits much shorter characteristic times than the electro-FORMING process [11, 14, 25] described in Section 2.1. A description of the FORMING process, in terms of the two main stages of dielectric breakdown is seen in Fig. 2-27. The initial leakage stage when T.A.T. (Trap Assisted Tunneling) dominates is followed by a hard breakdown region where drift diffusion is more dominant due to the large density of defects being generated during the thermal runaway.

The properties of the initial conductive filament (CF) created by the forming process strongly affect the device switching characteristics. Therefore, consideration of the CF must start with consideration of the FORMING process. A conductive filament generally exhibits semi-metallic characteristics indicating that it is associated with the Hf rich/oxygen deficient region in the dielectric. Therefore filament formation is associated with the oxygen
expulsion from a ‘preferable leakage path portion’ of the dielectric region – the essence of the filament formation process. Inside this ‘preferable leakage path portion’ hafnium-oxygen bond breakage is the required initial step in the CF formation process followed by the diffusion of the released oxygen ions.

---

**Figure 2-27:** A description of the FORMING process in terms of the 2 main stages of FORMING. 1) The initial leakage stage when T.A.T. (Trap Assisted Tunneling) is dominant. 2) Hard breakdown region where drift diffusion is more dominant due to the large density of defects being generated during the thermal runaway. After the second main stage current compliance is met and the voltage across the ReRAM (as seen in the insert) drops to ~ 0 V.

The forming process (described in Fig. 2-27) can be approximated by two stages. Stage 1 is characterized by limited vacancy generation and dominated by trap-assisted tunneling conduction via pre-existing vacancies along the grain boundaries. This has been simulated using the multi-phonon trap-assisted tunneling approach [55] and supported by C-AFM and STM results [56]. **Using a conductive-AFM tip (C-AFM), topographical-depressions were linked with grain-boundary locations.** Different conduction
mechanisms were found along the grain-boundary (GB) regions indicated by different I-V distributions per region. These C-AFM results are summarized in Fig. 2-28 indicating the grain boundaries as 'preferable leakage paths' in ALD deposited HfO₂ films.

Figure 2-28: Summarization of a C-AFM study [21, 38] depicting different conduction mechanism across the grain boundary regions of an ALD-HfO₂-thin-film. The charge transport in polycrystalline hafnium oxide has shown to occur preferentially at grain boundaries (GBs).

Stage 2 is labeled in Fig. 2-27 as the **fast transient region** and is characterized by a large number of concentrated defects which creates a defect ‘sub-band’ of conduction below the conduction band. During this final, sub-nanosecond, thermal-runaway stage the vacancy-generation proceeds via a self-accelerated (positive-feedback) process [57]. The defects supporting the electron transport through the dielectric are the source of power dissipation and their presence locally changes the dielectric constant, thus affecting the
electric field distribution. A higher defect density at a given location leads to higher
temperature and electric field in the adjacent regions. The local bond breakage describes
the defect generation rate and is seen in Equation 1. This equation describes the Arrhenius-
type behavior of the defect generation rate on a ‘bond-strength’ energy barrier (\(E_A\)) which is
also a function of the local electric field (\(E_{ox}\)) and the polarization constant of the chemical
bonds for the given dielectric layer [52, 53].

\[
G = G_0 * e^{-\frac{(E_A - \beta|E_{ox}|)}{K_B T}}
\]  

(1)

The probability of generating new defects in the vicinity of the existing one is higher
than that at a random location, and it grows with the density of defects in the surrounding
region. At the moment when a critical number of defects are generated close to each other,
a magnitude of the local field (\(\beta E_{ox}\)) approaches that of the bond strength (\(E_A\)) while
current induced temperature (\(T\)) is also increasing due to a higher charge transport rate
through this defect cluster [55]. Under these conditions, the defect generation probability
increases sharply and triggers a positive feedback process of self-accelerated bond
breakage. The breakage of additional Hf-O bonds leads to the creation of additional oxygen
vacancies supporting the electron transport, which further increases the electron flow
through this dielectric (CF) region. Higher local current density in this CF region increases
the local temperature in the surrounding region, promoting the generation of new defects
nearby (\(G\) is indirectly proportional to \(T\), from Equation 1). As these ‘clusters’ of generated
defects form inside the CF region they directly lower the resistance along the conductive
path which causes a greater share of the applied voltage to drop across the adjacent
dielectric regions of higher resistance which also enhances the vacancy generation rate. The
resistance of the resulting filament is determined by the magnitude of the current allowed by the utilized setup (the current compliance level).

This **temperature-field driven process of vacancy generation** in the vicinity of a critical size defect cluster describes a run-away dielectric breakdown in terms of Weibull statistics.

Figure 2-29 illustrates the aforementioned theoretical model of HfO$_2$-based filament formation along with what has been theorized to occur during subsequent resistive switching. This figure was illustrated with respect to the resistive switching model described in [20, 21, 55]. In this figure the preferable leakage path is seen to be along the (red line) grain boundary due to an initial accumulation of positively charge vacancies (green circles) inside an (atomic layer deposited, $\sim$350$^\circ$C-annealing) HfO$_2$ film. These defects were introduced into the dielectric (MeO$_x$) layer from the top electrode which has a strong oxygen affinity (such as a Ti OEL layer). As previously described, applying a voltage potential increases the leakage-current (inducing heating) and electrical field, thus **FORMING** a conductive filament.

The subsequent **RESET** and **SET** switching (also seen in Fig. 2-29) is a function of oxygen ion and vacancy distributions. Understanding this process in more detail is a fundamental goal of this dissertation as there are a few conflicting theories as to how **RESET** occurs [20, 21, 24, 58-60]. The resulting conductive filament is calculated to produce a non-uniform temperature profile along its length during the **RESET** operation, promoting preferential oxidation of the filament tip (as seen in Fig. 2-29) [20, 21]. This is followed by a field-driven dielectric breakdown of the barrier during the **SET** operation which restores the filament to its initially low resistive state. These findings point to the critical importance of controlling the filament cross section during **FORMING** to achieve low power ReRAM cell
switching [20, 21] and is the fundamental theoretical ground-work for the subsequent chapters of this dissertation.

2.5.3 Initial Leakage Current: Trap Assisted Tunneling

The dominant charge transport mechanism during the initial leakage state is Trap-Assisted-Tunneling (TAT) which is assisted by oxygen vacancy defects. An analysis of the initial leakage current versus temperature (seen in Fig 2-30) shows an increasing current with temperature and indicates a trap-assisted-tunneling dominant mechanism before any conductive filament is present in the dielectric. This was described by a quantum-mechanical model including electron-phonon coupling and lattice relaxation [55, 57, 61].
Figure 2-30: Fresh state leakage comparison at different temperatures and 2 different device areas.

Figure 2-31: Leakage currents measured (symbols) and simulated (solid lines) at different temperatures on MIM capacitors for both stoichiometric HfO$_2$ and sub-stoichiometric HfO$_{2-x}$. Simulations were performed with the multi-phonon TAT model proposed in [20, 21, 55, 57, 61]
Figure 2-31 [62] compares the measured leakage currents (data points) vs. TAT simulations (solid lines) for two different MIM capacitors: 1) with stoichiometric HfO$_2$ and the other with sub-stoichiometric HfO$_{2-x}$. Figure 2-25 validates the TAT current mechanism. Despite the same dielectric physical thicknesses, the leakage currents across HfO$_{2-x}$ are several orders of magnitudes higher than that of the HfO$_2$ [62]. The temperature dependency (increasing leakage current with increased temperature) and extracted activation energies of the conduction processes in these stacks were also calculated to be very similar [62]. These results indicate that GBs (which cover only a few percent of the total device area [48]) have a much larger density of oxygen vacancies compared to stoichiometric HfO$_2$ films [20, 21]. This is expected due to oxygen gettering by the overlying of a Ti metal film, as seen in [63]. The vacancy density is thus expected to be greater nearer the Ti metal layer [27, 63]. This underlines the importance of GBs, which are the preferential locations for defect accumulation and generation, and can therefore be considered as conductive filament precursors. Thermodynamically (in terms of Gibbs free energy (Fig. 2-25)) Ti, for example, should not be able to take oxygen from HfO$_2$. However, due to the weaker grain-boundary bonds, oxygen at the GBs is preferentially scavenged by Ti (or Hf, or Zr). In the work presented in subsequent chapters, different metal layers such as Hf, Zr, and Ti are used between the top-electrode and the HfO$_2$, and are labeled as oxygen exchange layers (OELs). These metals are not the same, however, they do show similar ‘gettering’ properties which as been reported in previous literature [20, 21, 37, 38, 51, 55, 57, 61, 63-66].

In the case of amorphous-HfO$_2$ (in reality, HfO$_{2-x}$) where no grain boundaries are present thin films tend to have defective or weak bonds [67]. These films will switch with or without OEL layers present [63, 64]. In amorphous-HfO$_{2-x}$, molecular-dynamic simulations [62, 68, 69] show the dielectric may phase separate into stoichiometric HfO$_2$ regions and Hf-rich domains, thus supporting a charge transport process similar to that
along the GBs. This infers the validity of these T.A.T. GB-based simulations and also infers some similarities to the phase-change mechanism described in previous sections.

2.6 Discussion and Summary

This preliminary electrical and physical characterization work sets the stage for the following chapters. The impact of forming overshoot current on the subsequent reset current and cycling characteristics was investigated through varying of metal-oxide thickness, composition, and dielectric configuration. Figure 2-32 summarizes the effects of OEL/MeOx thickness ratio, device size and ambient temperature on the forming I-V curve.

Sub-stoichiometry has been defined as a key factor enabling ReRAM switching which suggests a defect-assisted dominated mechanism for HfO₂-based ReRAM. ReRAM filament formation was introduced in terms of dielectric breakdown theory and in the follow chapters a detailed analysis of filament formation characterization and control will be discussed.

Figure 2-32: Summary of the general trends seen in FORMING.
CHAPTER 3: Filament-Formation: Characterization and Control

3.1 Introduction

The HfO₂-based dielectrics are a strong candidate for ReRAM applications due to their promise of high switching speed and endurance [35, 41]; although significant efforts are still needed to optimize performance of this type of memory stack. Recently, bipolar switching of HfO₂ based ReRAM operating under a current of 25µA was demonstrated [35, 70]. The reduction of power consumption for scaled devices with a high density memory cell array, as seen in Chapter 1, is also critical [4, 6, 8, 71]. Thus to achieve both lower-operative-current-switching, and a high memory window (R_{OFF}/R_{ON}> 10): current overshoot must be mitigated [32, 60]; and, the high-resistive-state-current (HRS) must be reduced with current compliance. However, as operative current reduces so does the size of the conductive filament (CF), thus making stable high speed endurance performance a challenge.

Specifically, one of the most critical operations that define device switching characteristics is the filament formation process i.e. FORMING. This chapter focuses on the electrical characterization of filament formation with respect to the role of current, time (in terms of dV/dt), voltage, and temperature. In addition, an analysis of the requirements for stable low power switching is presented. A systematic study on the dependence of the HRS-current on the voltage sweep rate (dV/dt), compliance current (I_{COMPL}), and ambient temperature during FORMING is discussed; and as a result, a demonstration of 1 µA, +/- 1V bipolar switching and a 100× reduction of the high-resistance-state-current for W/Zr/HfO2/TiN ReRAM devices are reported. As a means of mitigating the progressive
reduction of filament size with $I_{\text{COMPL}}$ the ambient temperature during $\text{FORMING}$ was raised to 150 °C, which resulted in the formation of a more stable filament at low current (1 µA).

In terms of RESET and SET switching, trade-offs are identified between reduced operative-current and switching-speed, thus enabling high-endurance up to $10^8$ cycles at 20 µA/200 ns pulse widths. RESET and SET characterization was carried out using both $dc$ and $ac$ endurance methods.

Controlling the progressive reduction of filament size with reduced $I_{\text{COMPL}}$ is critical for stable low power switching. As $I_{\text{COMPL}}$ decreases, the $\text{FORMING}$ voltage and ambient temperature become increasingly important. A broad distribution of the $\text{FORMING}$ voltage ($V_F$), as usually seen in HfO$_x$-based ReRAM devices subjected to the voltage sweep $\text{FORMING}$, strongly correlates with the variability of the subsequent maximum $\text{RESET}$ current, $I_{\text{max}}$ [32-34, 63, 67]. And the maximum $\text{RESET}$ current is considered to be a good figure-of-merit for characterizing the effective cross-section of the conductive filament (CF) through the dielectric [72]. Higher CF cross-sections translate to higher $I_{\text{max}}$ values, indicative of higher power consumption during memory cell operation. $\text{FORMING}$ voltage can be controlled when filament $\text{FORMING}$ is carried out under a constant voltage ($V_S$) condition, $V_F=V_S$ [37]. Employing a constant voltage $\text{FORMING}$ (CVF) at low $V_S$ further enables a lower current compliance limit ($I_{\text{COMPL}}$). Both, in tandem, lead to smaller (more manageable) CF cross-sections [20, 21]. However, $\text{FORMING}$ under such low $V_S/I_{\text{COMPL}}$ conditions may result in the limited out-diffusion of O-ions [20, 21] and the creation of an unstable CF (see below and [72]). In addition, CVF using low $V_F$ takes a rather long time, rendering it impractical for large-scale fabrication. In this study, we introduce a ‘hot’ $\text{FORMING}$ technique, in which CVF is performed at elevated temperatures (100 °C, 125 °C, 150 °C and 175 °C); it significantly reduces time-to-$\text{FORMING}$ and enables $\text{FORMING}$ at extremely low $V_F$ (down to 1.1V tested in this study). This ‘hot’ $\text{FORMING}$ technique lowers
the $V_s/I_{\text{compl.}}$ ratio while improving device switching characteristics in terms of memory window and device-to-device variability.

### 3.2 Experimental

The HfO$_x$ based ReRAM structure used in this work is shown in Fig. 3.1. The ReRAM devices were fabricated using W and TiN electrodes and a 5nm ALD HfO$_2$ dielectric (TE/OEL/MeOx/BE $\rightarrow$ W/Zr/HfOx/TiN). A thin oxygen exchange layer (OEL) of zirconium metal was deposited on the HfO$_2$ prior to top electrode deposition which resulted in a certain degree of oxygen deficiency of the HfO$_2$ after the final post-metal anneal. The device I-V characteristics during FORMING, 1$^{\text{st}}$ RESET, and SET/RESET switching were performed in dc voltage-sweep-mode or CVF using a Keithely 4200 parameter analyzer. Endurance testing was performed in ac (and dc) mode with a pulse width from 30 ns to 200 ns using an Agilent 81110A pulse generator. Both dc and ac measurements were performed on ReRAM test structures that ranged in size between 50 $\times$ 50 nm$^2$ and 1x1 $\mu$m$^2$. ReRAM test structures were connected in series with an external, commercially-available n-channel MOSFET (1T-1R configuration) that maintained the compliance current during FORMING and SET (Fig. 3.1a). Alternately, an integrated-1T-1R testing setup was utilized. FORMING and endurance testing were all performed using a wide range of compliance currents, voltage sweep rates, and ambient temperatures ($T_o$). The latter ranged from 25 °C to 150 °C. Figure 3.1b shows an example of the real-time voltage pulse waveforms used for FORMING/RESET/SET/ for each device during ac endurance testing. Figure 3-2 shows the cabling/switch matrix/pulse-generator setup for external 1T1R; and Fig. 3-3 shows the integrated 1T-1R SETup. Qualitative control of the filament size during FORMING was regulated through monitoring of three electrical parameters – namely, resistance after FORMING (LRS=low resistance...
state current read at 0.1V), $I_{\text{max}}$ during 1st \textit{RESET}, and HRS current during switching. The latter was also read at 0.1V.
Figure 3-2: ac endurance bench test setup. This was created in SEMATECH FEP reliability lab and used to create pulse waveforms seen in Fig.1b).

Figure 3-3: Integrated-1T-1R cabling and device SETup.
3.3 Parameters affecting Filament Formation and the Low-Resistance (ON) State: Current, Voltage Sweep Rate, Voltage and Temperature

Figure 3-4 illustrates the repeatability and switching control challenges intrinsic to low-power ReRAM switching, and, likewise, the motivation for a more fundamental understanding of the parameters affecting filament formation. Figure 3-4a shows high current ($I_{\text{COMPL}}=100 \, \mu\text{A}$) stability; and low-current (8 $\mu\text{A}$) instability. Figure 3-4b plots $I_{\text{max}}$ 1st $\text{RESET}$ vs. LRS-Resistance after $\text{FORMING}$. There is a linear trend seen between $I_{\text{max}}$ and LRS (Fig. 3-4b) until the current drops below $\sim10 \, \mu\text{A}$. The current-compliance levels seen in Fig 3-4a (100 $\mu\text{A}$ and 8 $\mu\text{A}$) match the $I_{\text{max}}$-1st-$\text{RESET}$ currents plotted in Fig. 3-4b (y-axis). This data was taken at room temperature with a 1T1R testing setup (which limited the current overshoot: explained in Chapter 2). Device sizes for this Fig. and the entire 3.3 Section are 1×1 $\mu\text{m}^2$.

Typical I-V characteristics of 1 $\mu\text{m}^2$ devices during $\text{dc FORMING}$ at various $I_{\text{COMPL}}$ levels are shown in Fig. 3-5a. Different slopes of the back-sweep curves (after $\text{FORMING}$) show that device resistance decreases with $I_{\text{COMPL}}$ and thus indicates a CF size increase. Dependence of device low-resistance-state (LRS) on $I_{\text{COMPL}}$ is shown in Fig. 3-5b. The maximum current observed during the very first $\text{RESET}$ cycle (1st $\text{RESET}$ $I_{\text{max}}$) is another indicator of the CF size. As shown in Fig. 3-6a, $\text{RESET} \, I_{\text{max}}$ linearly increases with $I_{\text{COMPL}}$ during $\text{FORMING}$. This dependence is amplified by increasing the $\text{FORMING}$ sweep rate i.e. $dV/dt$. Figure 3-6b shows that progressively reducing the $\text{FORMING}$ $dV/dt$ causes the subsequent 1st $\text{RESET}$ $I_{\text{max}}$ to approach its minimum value (equal to its respective current compliance during the previous $\text{FORMING}$ step, which is indicated by the solid lines). Thus, based on the data in figures 3-6a and b, it can be concluded that in order to achieve the minimum possible size of the CF, both the compliance current and voltage sweep rate during
**FORMING must be substantially reduced.** Decoupling the voltage sweep rate and the actual **FORMING** voltage will be shown in section 3.6.

![Figure 3-4: Plot of I_max 1st RESET vs. LRS-Resistance after FORMING.](image)

This figure indicates the problems with low-power switching. There is a linear trend seen between I_max and LRS until the current drops below ~10 μA. This data was taken with 1T1R testing setup and at room temperature.

To better demonstrate the effect of the **FORMING** voltage sweep rate on filament-formation; Fig. 3-7 compares the impact of both small (Fig. 3-7a) and large (Fig. 3-7b) dV/dt. Both **FORMING I_{COMPL}** levels are **SET** to 1.4 μA. At low dV/dt the CF size was not stable after **FORMING** as indicated by 1st **RESET** I_max being lower than **FORMING I_{COMPL}** and thus causing LRS and HRS values to be indistinguishable. In contrast, at very high voltage sweep...
rates during FORMING, the CF becomes too conductive to provide a notable memory window due to a drastically increased \( \text{RESET} I_{\text{max}} \) which results in a HRS current equal to the LRS-current (see red SET curve sub-plot in Fig. 3-7b).

Figure 3-5: (a) FORMING, in dc mode, for 1 \( \mu \text{m}^2 \) devices at various \( I_{\text{COMPL}} \) levels. In SET: \( I_{\text{COMPL}} = 2.5 \, \mu \text{A} \). (b) ReRAM resistance after FORMING as a function of \( I_{\text{COMPL}} \) during FORMING. CF size increase is schematically illustrated. Note: the ‘green’ x was taken at -0.1 V read voltage and yields the same value at the ‘red’ x taken at 0.1 V read voltage, seen in Figure 3a).
Data in Fig. 3-8a confirms that the CF size is insufficient to notably decrease the device resistance during FORMING at a low voltage sweep rate and low $I_{\text{COMPL}}$ of 1 µA (see black curve with star symbols). Increasing the FORMING $I_{\text{COMPL}}$ (green and blue colored curves in Fig. 3-8a) reduces the $dV/dt$ rate required for FORMING. To increase the filament size, and maintain a low FORMING $I_{\text{COMPL}}$, the voltage sweep rate must be increased ($dV/dt \sim 2V/s$) as shown in Fig. 3-8a. Note the black curve with star symbols which indicates the drop in resistance when FORMING $dV/dt > 2V/s$. Maintaining a low FORMING $I_{\text{COMPL}}$ of 1 µA, and substantially lowering $dV/dt$ can also be achieved by increasing the ambient
temperature during FORMING. This is illustrated in Fig. 3-8b wherein the red circles denote a decrease in required FORMING $dV/dt$ rate ($\sim$0.6V/s) when compared against the same room temperature 1 µA $I_{COMPL}$ data from Fig. 3-8a.

Using the optimized FORMING and 1st RESET conditions which were derived from evaluating the effects of ambient temperature and voltage sweep rate, stable $dc$ switching at room temperature was demonstrated (Fig. 3-9a) and at 150 °C (Fig. 3-9b) for the lowest operation current of 3 µA and 1.4 µA, respectively for this device size of 1×1 um$^2$. 

Figure 3-8: LRS resistance as a function of voltage sweep rate, $I_{COMPL}$ and temperature during FORMING.
3.4 High-Resistance (OFF) State Dependence on Max FORMING Current

As shown above, the maximum current observed during 1st \textit{RESET} is an effective indicator of the CF capacity – or more simply, size. The HRS-current observed during \textit{SET}, immediately after 1st \textit{RESET}, is the ultimate measure of a completely switchable filament – meaning it must be low enough to be distinguishable from LRS-current and create a memory window ≥ 10. Figure 3-10a compares 2 devices of the same 1×1 µm² size and formation parameters (dV/dt rate and \textit{FORMING} \textit{I}_{\text{COMPL}} (= 15 µA)). Keeping \textit{FORMING} sweep rate constant, Fig. 3-10a shows the 1st \textit{RESET} curves obtained after applying large (300 mV/s) and small (5 mV/s) 1st \textit{RESET} sweeping voltage rates. After the fast 1st \textit{RESET} \textit{dc} sweep, it is evident that the 1st \textit{RESET} \textit{I}_{\text{max}} was larger than the \textit{FORMING} \textit{I}_{\text{COMPL}} (= 15 µA). The insert (sub-plot) in Fig. 3-10a shows the resulting \textit{SET} characteristics. It is clear that a slower 1st \textit{RESET} results in lower HRS-current (blue curve) and a more stable R\textsubscript{OFF}/R\textsubscript{ON} ratio, whereas after a faster 1st \textit{RESET}, the HRS-current and LRS-current (red curve) are indistinguishable and, thus, do not provide a notable memory window. The observed ~100× reduction in HRS-current implies that the slower voltage sweeping rate during the 1st \textit{RESET} promotes increased filament oxidation. (The ‘slow’ 1st \textit{RESET} data can be compared to the results seen in section 3.6 where ‘slowing down’ the entire \textit{FORMING} process by...
reducing the FORMING voltage (also see Chapter 4 simulations) produces similar results without having to perform a slow 1st RESET. The latter is an additional and impractical step in terms of time required to form the device.) Figure 3-10b compares the HRS-current to the I_{COMPL} during FORMING. This linear dependence on FORMING I_{COMPL} is a fundamental trait seen in all the ReRAM devices for this work and similar to other indicators of the CF size. Moreover, after FORMING was performed at an elevated temperature (150 °C) and a slow 1st RESET was applied, multiple dc SET/RESET cycles at an operative current of 8 µA (Fig. 3-11) were possible. This is a notable major improvement from the 8 µA switching seen in Fig 3-4a.

Figure 3-10: a) Increase in memory window size in subsequent fast SET cycling due to a decrease in IHRS after a slow 1st RESET. b) Dependence of I_s on I_{COMPL}; as measured after dc and CVS 1st RESET. Note: I_s is the IHRS during SET just before the hard-break-down point (V_s).
Figure 3-11: Fast \textit{dc (RESET/SET)} switching after elevated temperature \textit{FORMING} and a reduction in 1\textsuperscript{st} \textit{RESET} $I_{\text{max}}$. This reduction in $I_{\text{max}}$ 1\textsuperscript{st} \textit{RESET} resulted from a reduction in the \textit{RESET} sweep rate.
3.5 Summarizing Filament Characterization Traits

Current is a dominant driving force in CF filament formation when $I_{\text{COMPL}}$ is high (> 25 μA). Lowering the $I_{\text{COMPL}}$ current (< 10μA) better reveals the effects of voltage and ambient temperature on filament formation. To summarize the aforementioned effects on low-power filament stability in terms of the voltage sweep-rate (dV/dt) during FORMING
(not 1st \textit{RESET}), and ambient temperature during \textit{FORMING}, Fig. 3-12, shows data for 4 similar devices (under a low current of 8 \( \mu \text{A} \)). These data represent various combinations of high and low \textit{FORMING} sweep rates and ambient temperatures (25 °C and 150 °C). Figure 3-12a shows the respective dc-sweep \textit{FORMING} events for each case. Figure 3-12b shows the corresponding 1st \textit{RESET} events and Fig. 3-12c shows the subsequent room temperature \textit{40 cycle switching average}. The curves labeled ‘1’, in Figs. 3-12a,b,c, show results for low dV/dt room temperature \textit{FORMING}; while those labeled ‘2’ show high dV/dt room temperature \textit{FORMING}. Curves labeled with ‘3’ and ‘4’ denote both low and high dV/dt \textit{FORMING} but at elevated (150 °C) ambient temperature. Considering the ‘1’ curve in Fig. 3-12a, where the low voltage sweep rate combined with an ambient temperature of 25°C enabled a \textit{FORMING} voltage of 1.8V, the 1st \textit{RESET} \( I_{\text{max}} \) indicates the filament was insufficient in terms of cross-section (or conductivity) to promote stable operation. This also elucidates how a higher \textit{FORMING} voltage leads to more post-\textit{FORMING} recombination before 1st \textit{RESET}. The ‘2’ and ‘3’ datasets both exhibit overshoot effects (Fig. 3-12b) even though a 1T1R configuration was implemented in the testing setup. These devices (Fig. 3-12c) exhibited poor switching due to increased HRS and smaller memory windows (HRS/LRS ratios). In light of Fig. 2-11a, this implies that larger dV/dt enhances the effects of parasitic capacitances. However, Fig. 2-11 is not necessarily consistent with dataset ‘1’, which incorporated a larger dV/dt compared to dataset ‘3’ which did not exhibit overshoot. It is concluded that elevating ambient temperature during \textit{FORMING} to 150 °C has a major impact on filament formation, especially when it is noted that initial leakage-current increases with ambient temperature (Figure 2-30) with respect to the dominant trap-assisted tunneling mechanism [20, 62]. The different shapes of the 1st \textit{RESET} curves (in Fig3-12b for data\textit{SET}s ‘2’ and ‘3’) indicate a mechanism change as the ‘2’ device is already in a pseudo-HRS before the 1st \textit{RESET} dc-sweep is applied. This is attributed to post-\textit{FORMING}
relaxation and which is discussed in Chapter 4. Dataset '3' indicates that a large \(\frac{dV}{dt}\) combined with elevated ambient temperature during FORMING creates filament cross-sections that are too large or conductive \((I_{\text{max} \ 1^{st}} \ \text{RESET} > 8\mu A \ \text{I}_\text{COMPL})\) and which require more time for filament oxidation. In Fig. 3-12b the allotted time for filament oxidation is constant and thus the HRS-current seen in Fig. 3-12c is higher which reduces the memory window (see curve '3' in Fig. 3-12c). Dataset '4' in Fig. 3-12a shows that increasing the ambient temperature while reducing \(\frac{dV}{dt}\) enables a very low FORMING voltage of 1.1V, which subsequently permits a 1\(^{st}\) RESET \(I_{\text{max}}\) in the range of 8 \(\mu\)A and facilitates HRS-current values sufficiently low and distinguishable from the LRS-current. This is indicated by the large memory window (Fig. 3-12c, dataset '4').

At this point it is apparent that a more in-depth study, decoupling the effects of \(\frac{dV}{dt}\) and ambient temperature, is needed as to better understand and control low-power filament formation. In the next section a method of 'hot' FORMING is introduced as a means to improve the stability of low-power filament formation.

### 3.6 Hot FORMING Method to Improve Memory Window and Uniformity of Low-Power HfO\(_2\)-x based ReRAMs

In terms of defining the effect of elevated temperature on the intrinsic filament-formation breakdown point, voltage sweep FORMING was performed on many devices. As seen in Fig. 3-13, under higher ambient temperatures, the FORMING voltage \((V_F)\) is significantly reduced. This is primarily due to trap-assisted tunneling and the higher rate of the electron transfer between the initial leakage current traps (oxygen vacancies) at elevated temperatures leading to greater power dissipation associated with the electron trapping/detrappping, and to a more efficient generation of new oxygen vacancies at lower voltage values [20]. High temperature FORMING not only leads to better device-to-device
$V_F$-uniformity, it also improves uniformity of the resulting LRS and $I_{\text{max}}$-RESET, which are shown in Figs. 3-14a, and b, respectively.

As indicated in the previous sections, LRS-after-FORMING and the $I_{\text{max}}$ during the 1st RESET are both good experimental indicators of CF size. However, as was previously reported, parasitic capacitances associated with the ReRAM circuit may enable the FORMING current through the ReRAM cell to exceed the current compliance limit even when it is controlled by the access transistor. Likewise, increasing the ambient temperature affects the final filament size (in terms of approximate cross-sectional area).

Figures. 3-15a and b show representative, full I-V curves of high $I_{\text{COMPL}}$ (100 µA) at RT (after FORMING at RT and 150 °C, respectively). Employing high temperature FORMING reduced the intrinsic $V_F$ value low enough as to enable sufficiently small filament cross-sections and suppress overshoot (defined as $I_{\text{max}} > I_{\text{COMPL}}$) as shown in Fig. 4-3c. These results, as expected, are more strongly evident when lower current compliance limits were applied (Fig. 3-14b).

![Figure 3-13: Voltage-sweep-FORMING preformed at150 °C and room temperature (RT). Each point represents a different tested device.](image)
Figure 3-14 a) LRS resistance (after-FORMING) extracted at -0.1 V during the 1st RESET. (b) Maximum RESET current during the 1st RESET. In both figures (a) and (B) 20 µA and 100 µA operational current levels were compared to show low and high power differences, respectively.
Figure 3-15 The 1st RESET and subsequent I-V characteristics after the voltage sweep FORMING performed at (a) RT/I\textsubscript{COMPL} = 100\,\mu A; (b) RT/I\textsubscript{COMPL} = 20\,\mu A; (c) 150 °C/I\textsubscript{COMPL} = 20\,\mu A.

Figure 3-16 shows the dependence of FORMING voltage on voltage sweep rate. This data indicates the need to separate the dV/dt dependence in order to more clearly understand the effect of FORMING voltage on CF formation. To decouple the effect of temperature on the overshoot from that of the FORMING voltage, FORMING was carried out under a CVF condition at different voltages and temperatures. These data are shown in Fig. 3-17a. The I\textsubscript{max} values exhibit a binary-like distribution. At each CVF condition a certain portion of the devices experienced overshoot. The fraction of devices that experienced...
overshoot increased with higher $V_F$ and higher temperature. In the devices with overshoot, the magnitude of the current overshoot ($I_{\text{max}}$) decreased with the FORMING voltage (Fig. 3-17b) since

$$I_{\text{max}} = I_{\text{c-forming}} + C_p \frac{dV_F}{dt},$$

where $C_p$ is the parasitic capacitance and $dV_F/dt$ is the voltage transient during the thermal (fast) runaway phase of the FORMING process (dielectric breakdown). The overshoot is also seen increasing with temperature, Fig. 3-17c, in accord with statistical simulations [37] based on the model referenced in [20, 21] and also simulated in Chapter 4. From the data in Figs. 3-17a, b and c, it is evident that the overshoot can be eliminated completely when FORMING is carried out at sufficiently low voltages (CVF between 1.1V-1.2V) even when CVF temperature is high (Fig 3-17b).

![Figure 3-16: Dependence of $V_F$ on $dV/dt$ illustrating the need for CVS FORMING and highlighting the need to decouple effects of $dV/dt$ and FORMING voltage on CF filament formation.](image-url)
Fig. 3-17 (a) $I_{\text{max}}$ values following constant voltage FORMING (CVF) performed at different voltages ($V_F$) and temperatures. Each symbol corresponds to a single device. (b) $I_{\text{max}}$ dependence on the FORMING voltage for different CVF groups performed at 150 °C. (All data taken from (a)). (c) An example of $I_{\text{max}}$ dependency on the CVF temperature. Each symbol in (c) represents an average of all devices in CVF group (1.2 $V_F$ from (a)) over a range of different temperatures.

On the other hand, the time-to-FORMING was drastically reduced at higher FORMING temperatures (Figs. 3-18a, b, and c) due to the exponential dependency of the vacancy generation rate on temperature, as well as applied voltage [20, 21]. By comparing characteristics of the RT and 150 °C CVF devices, which exhibit similar time-to-FORMING values (Figs. 3-19a and b), one can conclude that the hot FORMING method leads to narrower distributions of the LRS and HRS resistance values during subsequent room temperature switching and leads to increased cycle-to-cycle uniformity. 'Hot' FORMING with reduced $V_F$ (in Fig. 3-19a, 150 °C, CVF 1.2 V) results in more resistant (having smaller...
effective cross section) conductive filaments with better memory window (MW = LRS/HRS ratio, Fig. 3-19c). As shown in Fig. 3-19c, hot FORMING improves the uniformity of both the LRS and HRS components of the MW.

Fig. 3-18 Time-to-FORMING for CVF performed at different voltages at (a) 150 °C (b) 100 °C and (c) RT.
Fig. 3-19 Distributions of the (a) LRS and (b) HRS resistance values for devices formed under 150 °C/1.2 V and RT/1.8 V CVF conditions resulted in similar time-to-FORMING values. Each symbol represents a resistance value averaged over 15 consecutive SET/RESET cycles at RT. (c), Memory window of the devices formed at (a) 150 °C/1.2 V CVF and (b) RT/1.8 V conditions.

This proposed ‘hot’ FORMING method is shown to result in lower $I_{\text{max}}$ values and improved device-to-device and cycle-to-cycle uniformity of the LRS and HRS resistances. It also increased corresponding device memory windows. Constant voltage FORMING performed at low voltages, which is practically feasible when done at elevated temperatures (to reduce time-to-FORMING), helps to mitigate the negative effects of the overshoot caused by parasitic capacitances in ReRAM circuits.
3.7 SET and RESET Characterization: Low-Power Endurance Performance

3.7.1 Endurance testing: dc at various temperatures

The impact of ambient temperature on SET-LRS-resistance is demonstrated in Fig. 3-20. In Fig. 3-20a two devices are compared: one at low current compliance and the other at high current compliance. Both devices (starting at room temperature) were cycled 10 times with increasing ambient temperature. The high power (current compliance, ~ 100 µA) device shows no temperature dependence while the low power (current compliance, ~ 10 µA) device LRS resistance decreases with temperature. To further investigate the effect of ambient temperature on low power switching, Fig. 3-20b, shows cycling results of two low powered devices: (blue) cycled from room temperature to 150 °C in 10 cycle increments; and, (red) cycled the same number of times at each temperature but initiated at 150 °C and cycled down to room temperature. This data is re-plotted in Fig. 3-20c in terms of standard deviation (σ) for each cycled temperature. Note that that the (red) and (blue) arrows in Fig. 3-20b and Fig. 3-20c indicate the direction of the temperature change. Blue-triangles denote cycling down to 25 °C from 150 °C. Red-circles denote cycling up from 25 °C to 150 °C. Figures 3-20d and 3-20e show the actual device IV curves generated from this experiment. In Fig. 3-20d cycling becomes more stable with increasing temperature indicated by the black IV curves which overlay previous cycles and highlight the latter cycles at higher ambient temperature. As seen in Fig. 3-20c (red-circles) the cycling started at 150 °C remained stable as the temperature was decreased. Two important conclusions can be made from this low-power cycling data: 1) FORMING at 150 °C resulted in stable switching with less LRS resistance variation as compared to devices formed at room temperature; and 2), increased SET/RESET operational temperature resulted in the progressive reduction of the LRS resistance and, hence, progressively increasing the CF size.
This data confirms that temperature plays an important role in stable CF formation at low $I_{\text{COMPL}}$ levels. The impact of ambient temperature on SET-HRS resistance is demonstrated in Fig. 3-21, which shows a slightly increased HRS-variation with temperature.

The effects of ambient temperature are summarized in Fig. 3-22 for the high power case only (and shown for completeness with the low-power results of Figs 3-20 and 3-21). In Fig. 3-21 the high power LRS shows, as expected, little dependence on temperature, while HRS magnitude and variance both increase with temperature. This indicates that LRS and HRS are dominated by ohmic and T.A.T. conductance, respectively.
Figure 3-20 (a) Two devices are shown each with an I_{COMPL} of 10µA and 100µA, respectively. Each plotted point represents the mean low-resistance-state (LRS) value of 10 cycles at each respective temperature, both devices starting cycling at 25°C after FORMING at the same I_{COMPL} as SET. (b) Shows 2 more devices (both at low-power). Each LRS value for each cycle is shown. One device (after FORMRING) was cycled down from 150°C, 10 cycles at each respective temperature. The other device (after FORMRING) was cycled up from 25°C, 10 cycles at each respective temperature. (c) Shows standard deviation for the device cycles shown in (b). (d) I-V dc sweep plot of device data from (b). (e) I-V dc plot of device data from (b). All LRS read point were taken at 0.1V. These examples represent the data trends.
Figure 3-21: Shows $dc$ sweep endurance of an optimized device (after ‘hot’ FORMING) with an 8µA $I_{\text{COMPL}}$. This example device represents the data trend. Data shows HRS-current variation with temperature. Both temperatures were applied to the same device starting with 150 °C and ending with room temperature.
Figure 3-22: Trends of LRS and HRS Read current at different (cryo) temperatures.

3.7.2 Endurance testing (ac characterization) at 150C and room temperature

Figures 3-23a and 3-23b summarize the improved results of dc low-power switching, while Figure 3-20c compares the impact of ambient temperature on ac endurance. In Fig. 3-23c room temperature ac endurance was evaluated on a device formed
at RT and cycled at room temperature and compares it with endurance characterized at 150 °C on a device formed at 150 °C. Devices formed and operated at room temperature demonstrated enhanced endurance reaching the target of $10^8$ cycles. In contrast, devices formed and operated at 150 °C revealed degradation after only $10^6$ cycles. On the other hand, high temperature device operation at low operative current demonstrated better stability of the LRS current as compared to RT operation. The pulse width used in the aforementioned endurance test was 200 ns. The choice of pulse width was limited by poor switching performance at low operative current for shorter pulses, while increasing the operative current enabled stable switching at shorter pulses.

Figure 3-23: ac endurance at 20µA operation current performed at RT and 150 °C
3.4 Discussion

This work has demonstrated 1 µA, +/-1 V bipolar switching and a 100× reduction of the high-resistance-state current of W/Zr/HfOₓ/TiN ReRAM devices. This was achieved by identifying the effect of current, voltage, and ambient temperature which allowed for superior control of the conductive filament formation. The trade-off between reducing $I_{\text{COMPL}}$ (operative current) and switching pulse width was identified, while demonstrating high switching performance up to $10^8$ cycles at 20 µA of operative current and 200 ns of pulse width. Although more experiments are needed for better understanding of the particular role of each of these parameters, it is clearly evident that they all contribute to the local filament temperature that controls the filament size. The recently developed models of temperature dependent filament formation [34, 55, 57, 61, 73] represent the local filament temperature as $T_{\text{CF}} = T_o + R_{\text{th}} * V*I$, where $T_o$ is ambient temperature, $R_{\text{th}}$ is thermal resistance, and $V*I$ is power. The impact of temperature on LRS resistance, in this work, implies that ambient temperature indeed plays a notable role in the stable CF formation, as demonstrated by enabling switching at low operative current. A ‘hot’ FORMING method was also proposed and evaluated which improved device-to-device and cycle-to-cycle uniformity of the LRS and HRS resistances, as well as increased corresponding device memory windows.

Understanding how and why ‘hot’ FORMING works, on a microscopic level is critical in terms of understanding the IV curve shape differences in Fig. 3-12b and for the completion of a ReRAM simulation model which can be used in future work as an optimization tool. This Chapter focused on fundamental experimental investigations of ReRAM operations. The next chapter will use these experimental results as ‘calibration’ parameters to create a simulation model that encompasses all the necessary components of
filament growth which are dependent on current, temperature and voltage. These include ion/vacancy pair generation, oxygen ion diffusion, and recombination.
CHAPTER 4: Modeling ReRAM FORMING and (RESET/SET) Switching

4.1 Introduction

ReRAM technology presents an attractive option for embedded non-volatile memory systems if its (cycle-to-cycle and device-to-device) variability can be controlled. In the previous Chapter ‘hot’ FORMING was introduced as a method for controlling this variability. However, to optimize ReRAM performance further from the standpoint of both device structural properties and operation conditions it is necessary to understand the underlying physical processes responsible for these electrical characteristics. Causative connections between electrical measurements and physical properties can be effectively established through the use of simulations if explicit consideration of the HfO₂-specific nature of ReRAM characteristics is incorporated. In particular, relative oxygen affinities, valence states of ions and vacancies, ion diffusivities, and dielectric crystallinity must be quantitatively incorporated.

FORMING leads to the creation of an oxygen-deficient region, i.e. the conductive filament (CF) which is connecting the electrodes of the MIM memory device [20, 55, 57] and is extremely sensitive to the microscopic properties of the hafnium oxide film. In terms of operational simulation, two main regions of interest are identified: (1) the initial leakage stage which is characterized by limited vacancy-generation and dominated by trap-assisted-tunneling (T.A.T.) conduction via the limited number of pre-existing vacancies [52-54, 68]; and (2) the fast transient thermal-runaway stage (that has been documented to be in the sub-nanosecond time range) where vacancy-generation proceeds effectively via a self-accelerated (positive-feedback) process [20, 55, 56]. During this fast transient stage of
the vacancy-density within the preferable conductive path increases considerably, and the T.A.T. description as the dominate charge transport mechanism ceases to be valid. Thus, for the fast transient region one should consider the charge transport as electron drift through a created defect sub-band [55] as described in Chapter 2. The creation of the conductive filament requires that oxygen atoms be expelled which, in turn, requires breakage of the Hf-O bonds and the subsequent oxygen-ion out-diffusion. In other words, under an applied voltage and after the initial leakage stage, the conductive filament (CF) grows rapidly due to increasing amounts of oxygen vacancy generation and O\(^{-2}\) ion out-diffusion in a type of thermal runaway effect which leaves behind a highly conductive Hf-rich filament. Figure 4-1 describes this fast transient region (first described in Fig. 2-23) in more detail. These properties must be taken into account to properly model the FORMING operation. As discussed in Chapter 2 atomic layer deposited (ALD) HfO\(_2\) dielectrics tend to accumulate oxygen vacancies (V\(^{+2}\)) around their preferential leakage paths i.e. the grain boundaries [20, 52, 56]. And, in sub-stoichiometric HfO\(_{2-x}\) it has been shown that, due to a higher initial defect density, the effective activation energy for defect generation is reduced, thus causing a proportional increase in leakage currents and lowering the respective HfO\(_2\) dielectric breakdown strength [74]. In the case of sub-stoichiometry, HfO\(_{2-x}\)-based ReRAM this defect-assisted process of effective activation energy reduction has been shown to improve the impact of FORMING on the stability (RESET/SET) of subsequent switching [32, 63, 74].

In this work the FORMING operation is simulated for Hf-based ReRAM devices with the goal of identifying the main factors affecting operation variability and establishing general guidelines to optimize this key operation. More specifically, how and why does ‘hot’ forming work? It will be shown that this model has successfully reproduced, with no fitting parameters, the observed LRS and HRS trends with respect to dielectric stoichiometry, and
FORMING voltage ($V_F$). Through explicit consideration of the 3D vacancy distribution, it is shown that these simulations can be used to investigate the effects of different FORMING conditions on the overall filament geometry/composition. It is also shown that these simulations provide key insight with regards to the distribution of diffused-oxygen-ions in the surrounding oxide before \textit{RESET}, and how this distribution affects the conductive filament.

This simulation model also provided, for the first time to the best of the author's knowledge, a consistent explanation for all reported critical ReRAM features, including the role of O-deficiency (sub-stoichiometry) for repeatable switching. Discussion details include the relation between switching characteristics and dielectric morphology, and the advantages of inducing oxygen deficiency during device fabrication via the oxygen gettering process. The kinetics of \textit{RESET} gap formation based on the amplitude/duration of the \textit{RESET} input pulse are also described and concluded with preliminary \textit{RESET/SET} switching results which could lead to effective endurance and retention models with further development.
Figure 4-1: Schematic of the fast transient component of FORMING in terms of increasing initial leakage current which leads to a critical density of vacancy traps in the bulk (accumulated near the preferred leakage path i.e. grain boundary) and induces an increase in power dissipation (labeled $P$ in Figure). Temperature increases with power as per the Fourier Heat Transfer equation which induces more defect oxygen out-diffusion and defect generation, recursively, increasing the initial leakage current.

Figure 4-2: Flow diagram of the simulation procedure.
4.2 Detailed Description of the Model and Simulation Approach

4.2.1 Model Description: Initial Input Conditions

In order to describe the impact of FORMING conditions and material properties on device operations, a full 3D model was created in MATLAB capable of accounting for the stochastic nature of the FORMING and RESET/SET processes. The simulation flow chart is shown in Figure 4-2 for the case of a 20×20 nm² device with the following material layers: TiN(1nm)/Ti(3nm)/HfO₂(5nm)/TiN(1nm), where the nm value in parentheses represents each layer’s respective thickness. The program defines material properties in terms of thermal conductance and resistivity for each layer. Exact values and layer dimensions are given in Fig. 4-3a, 4-3b, 4-3c and 4-3d.

A pseudo lattice structure is defined in each layer by cubic bins with lengths, widths and heights equal to the respective bonding lengths. In the case of monoclinic HfO₂ the bonding length was λ=2.5 Å. This pseudo lattice structure provides constraints for vacancy and ion XYZ positions, and the bins are also grouped into 0.5 nm sized cubes which are used to calculate local resistivity changes with respect to the total number of defects present (inside each group of bins), illustrated in Fig. 4.3e.

The initial vacancy distributions (and subsequently generated defects (V+/O⁻₂ pairs)) are randomly created no closer than the reported O⁻ ion jumping distance of λ [42]. And in Fig. 4.3b the distribution of initial vacancies (blue dots) are seen to be confined within a 0.5 nm radius which represents the most “favorable” conductive filament (CF) point between 3D grains [20, 75] in its simplest form. This simple form of initial vacancy distribution is defined accounting for the oxygen ‘gettering’ action of the metallic reactive layer (in this case, metallic-Ti) above the HfO₂ dielectric, which leads to a higher vacancy concentration (more blue dots) closer to the metal layer [20, 27, 75]. One of the programs
main purposes is to model ionic diffusion effects through the dielectric; therefore, this simple view is expanded to consider the details of **3D grains and grain boundaries**. Grain boundaries have been shown to have accumulations of positively charge vacancies and consequently be responsible for the leakage current through the HfO₂ film [20]. In addition, there is increasing evidence [42, 52, 56] that extended defects along HfO₂ grain boundaries and dislocations can significantly affect the electronic properties of devices because other charged defects and impurities tend to diffuse more readily along them in response to an electric field [42, 52, 56]. As preliminary work for this model, MATLAB code was created (see Appendix 4 for details) which randomly generates outlined-patterns of 3D grains and gain boundaries in terms of varying initial vacancy densities (black dots in Fig. 4-3c). Fig. 4-3c shows a top view example and Fig. 4-3d shows an illustration representing what was attempted in Fig. 4-3c, noting that in Fig. 4-3b these densities are not shown for increased clarity of the conductive filament. A grain radius of ~ 5 nm and a grain boundary thickness of ~ 5 Å are seen in Fig. 4-3 and are easily adjustable. Considering that the most conductive GB region has the highest probability of conversion to a conductive filament during forming [62, 75], a center point between grains (illustrated in Fig 4-3d) is chosen as the most “favorable” conductive filament (CF) point between grains. This CF ‘grows’ with increasing amounts of generated defects as a function of effective activation energy reduction along the CF edges.
4.2.2 Model Description: Kinetics

Bond breakage and oxygen ion diffusion are simulated to match the reported behavior of oxygen ion diffusion inside hafnium-oxide meaning a defect pair is randomly generated in the HfO\textsubscript{2} (see next paragraph for details) initially spaced between lattice points (a distance 0.5*λ apart) so as to represent the first interstitial diffusion jump [76]. Subsequent diffusion jumps are reported to be substitutional and thus have a minimum jumping distance of λ [76].
Starting from the initial vacancy distribution, the program performs the computational loop in Fig. 4-1, which is repeated until \( I_{\text{COMPL}} \) is reached. At each loop step, the temperature-dependent resistivity, resistance, voltage-potential, power-density, current, and temperature in each XYZ grouped bin are calculated by numerically solving self-consistently Fourier's heat-transfer and the charge continuity equations. These 3D 'matrix maps' are then used as inputs for a kinetic Monte-Carlo model [77] which generates temperature/E-field-driven vacancy-ion pair generation (bond breakage), recombination, and oxygen-ion diffusion. The numerical method, which is first compiled in C-code and then embedded into MATLAB, is a variant of the Gauss–Seidel method which uses successive over-relaxation to solve linear systems with a faster convergence [78] (see Appendix 3). The statistically-driven kinetic Monte-Carlo portion of the code simultaneously calculates the rates (Hz) associated with each possible 'event' i.e. generation, recombination, and diffusion. This process depends on 2 random number inputs which respectively account for both indexing the next event in terms of its probability and the sum of competing events. For example, if 1,000 events were all competing to be the next event each with its own generation rate the lifetime between occurring events should increase by 3 orders of magnitude.

The rates associated with the vacancy generation (G), ion diffusion (D), and recombination (R), are shown in equations 1 through 3.

\[
G = G_0 * e^{-\left(\frac{E_G - \beta |F_{\text{ext}}|}{K_b T}\right)} \\
D = D_0 * e^{-\left(\frac{E_D - Q_2^2 (F_{\text{ext} + F_{\text{CR}}})}{K_b T}\right)} \\
R = R_0 * e^{-\left(\frac{E_R}{K_b T}\right)}
\]

(1) (2) (3)
Here, the characteristic vibration frequencies \( G_o, D_o, \) and \( R_o, \) are all equal (see Table I) as described in [50, 55, 61, 79]. \( K_b \) is Boltzmann's constant, and \( E_G, E_D, E_R \) are the activation energies of the corresponding processes. \( Q \) is the charge state of the ion, and \( \lambda \) is the oxygen ion hopping distance corresponding to the next-neighbor oxygen lattice sites. \( F_{ext} \) is the electric field induced in each local site by the externally applied voltage, and \( F_{CR} \) is the local field determined by the oxygen ions and vacancies charge states. Note that \( (E_G - \beta [F_{ext}]), \) and \( (E_D - Q^{\lambda \over 2} [F_{ext} + F_{CR}]) \) are the effective-activation-energies at each XYZ location. The zero-field activation energy \( E_G \) and the bond polarization factor \( \beta \) [53] of the vacancy generation process were extracted from TDDB measurements of the TiN/Ti/HfO_\text{x}/TiN capacitors using the method in [79, 80]. As seen in Table I, the dielectric constant, \( k, \) is considered to be higher inside the GB region due to a higher density of non-passivated metal bonds in this region. Due to a significant energy gain associated with the vacancy-ion recombination process, its activation energy is expected to be smaller than \( E_D \) (see Table I).

As previously described, the \( O^{2-} \) ion diffusion in the lattice (see Eq. (2)) was simulated to proceed substitutionally (following its initial shift from a regular lattice position to the interstitial one) with an activation energy \( E_D = 0.7\text{eV} \) [42]. Values for the parameters used in the simulations are reported in Table I and are consistent with previous studies [56, 61]. At every loop step, the Monte Carlo method is used to determine the most probable 'event', i.e. oxygen vacancy generation, oxygen ion diffusion, and vacancy-ion recombination. The ion and vacancy 3D distribution maps are then updated accordingly.
4.3.1 Determining the Resistivity of Sub-Stoichiometric Hafnium-Oxide

In this study, simulation efforts are focused upon the final thermal-runaway phase as a means to build upon previous work of characterizing the T.A.T. dominated initial leakage stage of FORMING, and using an effective-resistance (ohmic) approach to describe the electron-transport. At any given (simulation-time) moment, the current through the ReRAM device is calculated by solving the network of these elementary resistors constituting the conductive path. The resistance (\( R = \rho \times \text{Length}/\text{Area} \)) of each XYZ bin is derived from the respective resistivity, \( \rho = \rho_0 (1 + \alpha \times \Delta T) \), where the non-temperature-dependence resistivity, \( \rho_0 \), is first extracted from the defect concentration (see Fig. 4-4a, b), \( \alpha = 0.0037 \) is the coefficient for temperature dependence of the electrical resistance, and \( \Delta T \) is the temperature difference from room temperature. The relation between the vacancy density in each XYZ group-of-bins and its resistance was estimated based on a Hf-metal oxidation study [46] where decreasing oxygen flow rate (sccm) corresponded to an increasing amount of stoichiometric change. The process of extracting the effective defect density, finding the corresponding stoichiometric change (2-x) and relating it in terms of resistivity is described in Fig. 4-4a, b. This effective resistance model allows for the description of filament growth during the thermal runaway phase and also the investigation of FORMING conditions on CF geometry as well as on O\(^2\) distribution after FORMING, which are expected to affect the subsequent RESET process.

**TABLE I**
MAIN PARAMETERS USED IN SIMULATIONS

<table>
<thead>
<tr>
<th>HfO(_2)</th>
<th>( E_G ) [eV]</th>
<th>( \beta ) [eÅ]</th>
<th>( k )</th>
<th>( E_D ) [eV]</th>
<th>( E_R ) [eV]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulk</td>
<td>6.2</td>
<td>40</td>
<td>21</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>GB</td>
<td>2.8</td>
<td>40</td>
<td>30</td>
<td>0.7</td>
<td>0.5</td>
</tr>
</tbody>
</table>

\( G_0, D_0 \) and \( R_0 \) where taken at \( 7 \times 10^3 \)Hz.
A) Determining Stoichiometry in terms of Defect Density

Find defect concentration of each point inside ReRAM (Normal Distribution of $V^{x0}$)

$$f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Normalize to volume (group-of-bins size)

$$Defect\_concen = \frac{f(x)}{\sum f(x) \cdot group\_of\_bins\_Volume}$$

Use molecular density $^*$ of HfO$_2$ to determine proper stoichiometry

$$x = \frac{Defect\_concen}{n_0(Density_{HfO_2} / Molar\_Mass)} = \frac{Defect\_concen}{molecular\_density_{HfO_2}} [\text{cm}^{-3} \frac{1}{g/cm^3 \text{ mol}^{-1}} \rightarrow \text{cm}^{-3} \frac{1}{\text{mol cm}^3 \text{ mol}^{-1}} \rightarrow \text{cm}^{-3} \text{ mol}^{-1}]$$

normalizing the number of vacancies in the ReRAM to be within the limits of allowed atomic spacing

**Stoichiometry change in HfO$_2 = 2 - x$**

$^*$ ratio between the mass of one mole of a substance and its volume

B) Extracted Resistivity: HfO$_{2-x}$

Decreasing oxygen flow rate, corresponds to increasing $x$, HfO$_{2-x}$

Source: Erwin Hildebrandt, et. al.

$\rho_{HfO_2-x} = 1e5 \ (\Omega \cdot \text{m})$

$\rho_{Hf} = 4e-5 \ (\Omega \cdot \text{m})$

Model Estimation**

$$\rho_{(ReRAM)} = (\rho_{HfO_2} + C_1) * e^{-(C_2 - (2-x))^4} + \rho_{Hf}$$

** same trend for thermal conductance ($k_o$)

Figure 4-4 (a) Determining the stoichiometry from the ion/vacancy positions in the 3D matrix space. (b) Estimation of the relation between the vacancy density in each XYZ bin and its experimental resistance values.
4.3 Modeling the Effects of Different FORMING Conditions on Conductive Filament Stability

4.3.1 Simulated Filament Properties vs. Different FORMING Conditions: Comparing Different $V_F$ and Temp. Conditions on vs. Oxygen/Vacancy Distributions

To delineate the effects of the forming voltage and ambient temperature on filament geometry and oxygen ion distribution, simulations of the forming process were carried out for different high and low forming voltages ($V_F=2.5$ V, 1.7 V) and ambient temperatures (T= 25 oC, 150 oC) for the same current compliance limit of 20 $\mu$A (Figs. 4-5 and 4-7).

Figure 4-5 shows oxygen ion/vacancy ($O^{-2}/V^0$) distributions for the ‘Immediately-After-FORMING’ thermal runaway phase. Figures 4-5a, 4-5e, 4-5i, and 4-5m display the 3D maps of the $O^{-2}/V^0$ distribution for each respective FORMING condition at the point when the applied voltage is turned OFF (red, white and blue colored dots respectively represent $O^{-2}$ ions inside HfO$_2$ bulk dielectric, $O^{-2}$ ions inside the Ti OEL layer, and $V^0$ inside the CF). Figures 4-5b, 4-5f, 4-5j, and 4-5n represent a side view of each respective filament geometry in terms of resistivity (Dark red-HfO$_2$ = 1e5 $\Omega$-m, Blue-Hf = 4e-5 $\Omega$-m). For clarity, Figs. 4-5c, 4-5g, 4-5k, 4-5o, and 4-5d, 4-5h, 4-5l, 4-5p show the radial and vertical distributions of $O^{-2}$ ions and vacancies. By comparing the Immediately-After-FORMING radial and vertical distributions, in Fig. 4-5, one can see that under high $V_F$ (= 2.5 V) conditions the ion/vacancy distributions approximately overlap each other indicating limited $O^{-2}$ diffusion. As indicated by the blue and black curves in Figs. 4-6a, and 4-6b, this is accompanied by limited amounts of recombination events during the FORMING fast-transient growth stage. This is strikingly different from the low $V_F$ (=1.7V) radial distributions in Fig. 4-5, which exhibit more spread-out ion/vacancy distributions, and a relatively larger concentration of recombination events during the FORMING fast-transient growth stage indicated by the red and green curves in Figs. 4-6a, and 4-6b. Figure 4-7 shows the vacancy/oxygen ion distributions for the same FORMING conditions as Fig. 4-3 just after a period of relaxation.
$(\text{Time} \geq 1 \mu s)$ with voltage and current both equal to zero. For simplicity, the voltage is zero during this Post-FORMING Relaxation, although a certain voltage may still be present across the memory cell due to a voltage-division effect between the ReRAM and the external resistance as the voltage is ramped down. Comparing the Immediately-After-Forming simulation results of Figs. 4-5a, 4-5e, 4-5i, and 4-5m with Post-FORMING Relaxation (Figs. 4-7a, 4-7e, 4-7i, and 4-7m) it can be determined that the $O^{2-}$ ions are still diffusing when both voltage and current equal zero. This is assisted mostly by Coulomb repulsion between closely neighboring $O^{2-}$ ions, Eq. (2) which induces an increase in post-FORMING recombination for this low-power (20 $\mu$A) $I_{\text{COMPL}}$, which is plotted in Figs. 4-6c, 4-6d).

Vacancy/Oxygen ion distribution Immediately After Forming

Figure 4-5: Simulation results regarding the effects of the FORMING voltage amplitude and ambient temperature (as specified on the graphs) on the resulting filament geometry and $O^{2-}$ distributions at point that $I_c$ is reached. Sub-figures (a,e,i,m) show 3D positions of vacancies (blue dots) and oxygen ions (red dots, white dots); Sub-figures (b,f,j,n) show 3D resistivity maps (Dark red-HfO$_2$=1e5$\Omega$-m, Blue-Hf=4e-5$\Omega$-m); Sub-figures (c,g,k,o) show ion and vacancy radial distribution in terms of concentration per cubic nanometer. Sub-figures (d,h,l,p) show ion and vacancy vertical distributions in terms of concentration. The ‘Time’ represents the total time simulated for the thermal runaway phase of FORMING.
Figure 4-6: Concentration of recombination events 'Immediately After FORMING' (from Fig. 4-5) and 'After Relaxation' (from Fig. 4-7). Radial (a, c) and vertical (b, d) distributions of ion-vacancy recombination events.
Figure 4-7: Simulation results regarding the effects of the FORMING voltage amplitude and ambient temperature (as specified on the graphs) on the resulting filament geometry and $O^{2-}$ distributions after a period of relaxation, i.e., no applied voltage for a time $\geq 1\,\mu s$. Sub-figures (a,e,i,m) show 3D positions of vacancies (blue dots) and oxygen ions (red dots, white dots); Sub-figures (b,f,j,n) show 3D resistivity maps ($\text{Dark red-HfO}_2 = 1\times10^5\,\Omega\cdot\text{m}$, Blue-Hf = 4\times10^{-3}\,\Omega\cdot\text{m}$); Sub-figures (c,g,k,o) show ion and vacancy radial distribution in terms of concentration per cubic nanometer. Sub-figures (d,h,l,p) show ion and vacancy vertical distributions in terms of concentration. The 'Time' represents the allotted time for relaxation.

In comparing the distribution plots for this 20 $\mu$A low-power case-study, in Figs. 4-5, 4-6, and 4-7, there are some key notable features: (1) For the high voltage cases of Fig. 4-5 (under both 25 $^\circ$C and 150 $^\circ$C ambient temperature) there are more oxygen ions inside the CF region compared to the low voltage cases (this is summarized in Fig. 4-8 by the overlapping oxygen-ion/vacancy distribution curves for the $V_F=2.5\,\text{V}$ case); and (2) After relaxation (in Fig. 4-7), all ions that were inside the CF after FORMING have either diffused...
into the HfO₂ bulk or recombined depending upon on their respective starting positions and local temperature (Fig. 4-9a).

**Results: Differences in Oxygen/Vacancy Distributions After Forming**

<table>
<thead>
<tr>
<th>Oxygen/Vacancy Ion Distribution</th>
<th>Immediately-After-Forming</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vᵢ = 2.5V</strong></td>
<td><strong>Vᵢ = 1.7V</strong></td>
</tr>
<tr>
<td><strong>Ion / Vacancy overlap</strong></td>
<td></td>
</tr>
<tr>
<td>Generation Rate &gt;&gt; Recombination and Diffusion Rates</td>
<td>Generation Rate &gt; Recombination and Diffusion Rates</td>
</tr>
<tr>
<td>Radial Distribution</td>
<td>Radial Distribution</td>
</tr>
</tbody>
</table>

After Relaxation Period (Time ≥ 5μs; V, I = 0)

More Recombination | Less Recombination

**Figure 4-8:** Summary of low-power results in Fig. 4-5 and 4-7, indicating the differences in oxygen ion and vacancy distributions after two different Vᵢ FORMING conditions. Green arrows point to the different Side View and Resistivity maps after FORMING relaxation.

**a) Temperature Distribution**

**b) E-field Distribution**

**Forming condition: 2.5V/25°C**

Compliance = 20μA

**Figure 4-9:** a) An example of local filament temperature (a) and electric field (b) distributions at the beginning and end of FORMING under the 2.5 V/25 °C/20 μA condition.
As the filament was created during this fast transient thermal-runaway stage, the defect generation rates at multiple sites along the conductive path were higher than the ion diffusion and recombination rates. Irregular, local, variations of higher-conductive regions associated with the filament geometry combined with the charge of each local defect both define the magnitude and location of local field increase. As the filament ‘grows’ these fine features tend to smooth out the filament shape and reduce the resistivity non-uniformity in the filament region as seen in Fig. 4-9b. When the fast transient runaway phase of the FORMING occurs more rapidly, as in the high voltage cases (both 25 °C and 150 °C), the local field increase is high enough to lower the effective activation energy for oxygen vacancy generation below that of oxygen ion diffusion and recombination. Thus, a significant number of generated oxygen ions do not have time to diffuse away from the filament or recombine (Figs. 4-5k, 4-5l, 4-5o, 4-5p). As a consequence, a significant number of recombination events take place during the post-FORMING Relaxation phase (Fig. 4-6c and d and Fig. 4-8) leading to the formation of a less robust and less stable filament after high $V_F$ (= 2.5 V) with respect to the lower $V_F$ (= 1.7 V) conditions at both 25 °C and 150 °C. Figure 4-10a shows in more detail the difference between the low and high $V_F$ cases in terms of the increased current-variation during CF growth, and variations in peak E-field amplitude within the filament. Figure 4-10b also summarizes the CF growth ‘chain of events’ and indicates that a lower $V_F$ leads to improved low-power filament stability.
4.3.2 Resistance and Cross-sections of Conductive Filament

The effect of different FORMING conditions on the low resistance state (LRS) predicted by our model agree with the trends observed experimentally. Fig. 4-11a shows the comparison between LRS distributions measured and simulated after a FORMING operation performed at 100 µA and 20 µA I_{COMPL}. Experimental data were taken on a 1T1R ReRAM structure with the integrated transistor set-up to reduce overshoot. As can be seen, the reduced LRS variability observed at low $V_F$ is correctly reproduced by our simulations. 

This reduced variability is a direct consequence of the much lower post-FORMING recombination discussed above. The effective-filament-radius extracted from simulations is shown in Fig. 4-11b. This figure implies that when FORMING is performed at higher voltages
the final filament cross-sections are considerably reduced for the 20 µA I\textsubscript{COMPL} case due to post-FORMING recombination. On the other hand, the cross section of the 100 µA I\textsubscript{COMPL} case is negligibly affected. This also matches experimental data shown in Fig. 4-11a. Fig. 4-11b which also suggests that the CF size is mostly controlled by the current compliance and at low current compliance the CF size is more stable at lower V\textsubscript{F}. Figure 4-11b also indicates that ‘hot’ forming increases the CF size. This makes sense in terms of the dominance of the drift-diffusion during the thermal runaway which dictates a lower resistance with higher temperature. This would require a larger CF area with the same current compliance limit. The trends in Fig. 4-11b suggests that under the low-power (20 µA) current compliance limit, higher FORMING voltages leads to unstable filament formation and ‘hot’ FORMING increases the CF size.

4.3.3 Evaluating the Filament Stability under Low Current Compliance

The 3D resistivity maps for both Immediately-After-FORMING and Post-FORMING-Relaxation are correlated to match corresponding stages of the experimental I-V characteristic DC sweep curves in Fig. 4-12. The filament resistivity maps Immediately-After-FORMING thermal runaway, are shown in Figs. 4-12a, and 4-12c, and the filament resistivity maps Post-FORMING-Relaxation are plotted in Figs. 4-12b, and 4-12d.

In the case of high-V\textsubscript{F} FORMING (Figs. 4-12a, and 4-12b), the higher electric field decreases the time of the thermal runaway phase (when most of the vacancies are generated) thus limiting the time for the released O\textsuperscript{2-} ions to diffuse away from the filament. The significant post-FORMING ion-vacancy recombination (specifically, around the filament region near the bottom electrode) occurring in this case causes a partial re-oxidation of the filament, (Figs. 4-12b, and 4-12d). In the case of low-V\textsubscript{F} FORMING (Figs. 4-12c, and 4-12d) the filament maintains its low resistance nature through the relaxation phase, consistent
with the measured I-V data. For this low-power, 20 µA case-study our simulations qualitatively describe a drastic difference between the post-
FORMING conduction, ohmic and highly non-linear, in the devices formed under different voltages/temperatures. To summarize Fig. 4-12, high $V_F$ leads to a higher electric field which decreases the time-to-forming during thermal runaway phase; this limits the $O^2$ out-diffusion causing a significant amount of post-forming recombination. For the low $V_F$ case simulations indicate that there is substantially more $O^2$ out-diffusion during the CF growth which limits the subsequent post-forming recombination enabling a low-power stable filament that is more likely to produce RESET/SET switching.

4.3.4 Discussion of Filament Stability Results

Here, an investigation is presented regarding the factors controlling the conductive filament stability using a Monte-Carlo physics-based model to simulate the thermal runaway phase of the FORMING process within the effective resistance description of the charge transport in the conductive path. According to our experimental results, for 20 µA $I_{\text{COMPL}}$, a significant reduction of CF stability is observed under high FORMING voltage conditions. The simulations suggest that this is due to a local field increase which is sufficiently high to lower the effective activation energy for vacancy generation below that of the oxygen diffusion and recombination processes. As a consequence, a significant number of recombination events take place during the relaxation phase leading to the formation of a less robust and less stable filament after high voltage FORMING compared with low voltage FORMING conditions for both RT and 150 °C. ‘Hot’ forming increases the size of the CF cross-sections as well as lowering the $V_F$. Thus, it serves to increase the overall filament stability at low-power switching, which is consistent with the experimental results presented in Chapter3.
Figure 4-11: (a) Simulated vs. experimental distributions of the resistance values and (b) estimated values of the filament radii before and after the post-FORMING relaxation under different FORMING conditions (100 µA and 20 µA: 25 °C and 150 °C). The experimentally assessed [6, 14] effective resistivity value of 4e-5 Ω·m was used.
4.4 Connecting Physical and Electrical Properties of Hafnia-based ReRAM

4.4.1 Effect of Oxide Sub-stoichiometry on Oxygen Ion Movement after FORMING

To further analyze the effects of oxide sub-stoichiometry, it is necessary to consider grain boundaries in the ALD-grown dielectric film which were previously introduced as having accumulated positively-charged-oxygen-vacancies (black simulated dots) compared to the CF region where vacancies (blue simulated dots) are neutrally charged due to electron flow.

As indicated in the previous section the released oxygen ions, which remain in the filament's immediate vicinity, tend to recombine with the vacancies there. Thus, the final CF characteristics depend, not only on the current compliance value, but also on a spatial distribution of the released oxygen ions at the end of the forming process. The post-forming
Ion distribution is the critical factor affecting the efficiency of the subsequent reset process and is controlled by the morphology (i.e. crystallinity) of the dielectric material. Indeed, due to a very low $O^2$ hopping barrier (0.7 eV), the oxygen ion diffusion proceeds very effectively at room temperature which would lead to an eventual uniform distribution of expelled oxygen ions throughout the entire dielectric volume. This is shown in Figs. 4-13a through 4-13d which present a simulated example of the forming filament ‘growth’ process in a 20x20 nm TiN/Ti/5nmHfO2/TiN cell with no pinned vacancies. Figure 4-13a though 4-13c illustrates filament ‘growth’ during FORMING. Figure 4-13d shows the After-post-FORMING-relaxation with a simulated total time of .0004 seconds. As seen in Fig. 4-13d, this relaxation results in an oxygen-depletion of the dielectric region around the filament, thus, disabling the filament re-oxidation process which controls reset efficiency. Similarly, Figs. 4-13a’ through 4-13d’ show a second simulated device with pinned positive charges along the grain boundaries which attracts the negatively charged oxygen ions preventing them from leaving the CF region. The energy barrier associated with the vacancy segregation at the GBs [52] reduces the probability of recombination between the negatively charged ion and pinned positive vacancies. Therefore, these coupled oxygen ions remain within the limited region defined by the GBs surrounding the conductive filament. The after-post-FORMING-relaxation (seen in Fig. 4-13d’) for the same allotted total time of .0004 seconds confirmed that the pinned vacancies, with a density determined by the degree of the hafnia oxygen deficiency, can effectively contain oxygen ions. It stands to reason that this enables higher endurance of the HfO$_2$-based devices, and thicker RESET gap formation (described in more detail below). The next step is to consider different (low, medium, and high) vacancy densities as to better understand the effects of sub-stoichiometry on subsequent RESET/SET switching.
Figure 4-13. Two simulated device examples of the forming filament ‘growth’ process in 20x20 nm TiN/Ti/5nmHfO2/TiN cell, $V_F=1.7V$, $I_{COMPL}=50\mu A$. The red and white dots represent oxygen ions in the interstitial positions in the dielectric, and those which diffused into the Ti metal layer and oxidized it, respectively. Cyan dots (only seen in (d)) are the oxygen ions which diffused outside of the cell volume. Blue and black dots are oxygen vacancies generated during the forming process, and the as-processed pinned vacancies along the grain boundaries. (a-c) Indicate time snap-shots of filament growth for one simulated device. Neither pinned vacancies nor 3D grain boundaries are present in this simulation. (a’-c’) Indicate time snap-shots of filament growth for the second simulated device. Pinned vacancies and 3D grain boundaries are present in this simulation. (d and d’) Show after-post-forming-relaxation for both devices respectively. Voltage and Current both =0. Post-forming relaxation time was 0.0004 sec.

Fig. 4-14 expands on the experiment described in Fig. 4-13; the reader should note that this Figure has two separate parts for better clarity. Figures 4-14 through 4-17 analyze the effect of pinned vacancies on FORMING stability and subsequent RESET gap formation. Figures 4-14 (a1, and a2) through 4-14(d1, and d2) outlines the initial conditions of a
simulated 20 × 20 nm² device. It has been shown that sub-stoichiometry induced from the oxygen gettering layer at the overlaying metal layer (OEL) is known to dramatically increase the vacancy density in GBs [81]. Therefore, an arbitrary created grain boundary (GB) pattern is considered with four different levels of hafnia sub-stoichiometry (induced by OEL) resulting in different densities of the positively-charged oxygen vacancies at along the GBs (none – fully stoichiometric, low, medium, and high). Figs. 4-14 (a3 - d3) show the simulated forming I-V curves that were outputted from the initial conditions. Figs. 4-14(a4 and a5) though 4-14(d4 and d5) show the ion/vacancy positions after after-post-FORMING-relaxation. In the ‘side view’ Figs. 4-14(b4 - d4) all ‘pinned’ vacancies along the GBs were removed for clarity.

In comparing the two extreme cases of ‘no-pinned vacancies’ and ‘high pinned vacancies’, Figs. 4-14(a - d), it can be seen that at lower sub-stoichiometry conditions more oxygen ions diffuse into the OEL layer (white dots). The vertical distribution plots of Figs. 4-14(a6 - d6) confirm that the number of O² ions diffusing into the OEL increases with decreased initial vacancy density. It has been shown that when oxygen ions diffuse in the OEL layer they become consumed, oxidizing the Ti metal layer, and are no longer available for future operations in terms of RESET/SET switching [20]. On the other hand, in the ‘no-pinned vacancy’ stoichiometric case, it can be seen that the O² ions that remained in the bulk HfO₂ (red dots in Fig. 4-14a4) have diffused far away from the CF region (as expected). In the high pinned vacancy density case (red dots in Fi. 4-14 d4, and d6) the ions have not diffused far away; however, there are a limited number of them in total. Comparing the differences between these two cases will be important when analyzing the subsequent RESET results in Section 4.4.3. Before covering the detailed results of sub-stoichiometry on RESET for all 4 cases of Fig. 4-14, it is appropriate to first introduce the simulated RESET kinetics for the ‘medium pinned vacancy’ density case of Fig. 4-14c as a benchmark.
Initial Conditions: Distribution of Pinned Vacancies

Figure 4-14(PART1) (a1, and a2) through (d1, and d2): Initial conditions: 4 levels of Hafnia substoichiometry (induced by OEL) resulting in different densities of the positively-charged oxygen vacancies at GBs (none – fully stoichiometric, low, medium, and high) are considered.
Figure 4-14 (PART2) (a3 - d3) Simulated forming I-V curves. (a4, and 5) though (d4 and 5): Ion/vacancy positions after forming. In a4 –d4 all 'pinned' vacancies in GBs were removed from the figure for side-view clarity. (a6 though d6) Vertical distributions for the vacancies and oxygen ions after the forming event and following relaxation period when the applied voltage is removed. Reset for each one of these simulated devices is seen in Fig. 4-17 and 4-18.
4.4.2 Effect of Pulse Amplitude/Duration on RESET Gap Formation

During the RESET operation, the reversed voltage polarity drives oxygen ions located at the interstitial positions in the dielectric towards the bottom (‘RESET anode’) electrode, which needs to be a good oxygen barrier like, for instance, a stoichiometric TiN. (Those ions, which reached the (Ti) oxygen exchange layer during FORMING, form a metal oxide, which cannot be reduced by applied RESET voltages – this invalidates previous models used to describe metal oxide RRAM operation such as reported by Shimeng et al. [60]). Coulomb repulsion between the oxygen ions concentrated near the bottom electrode increases the density near the bottom of the conducting filament and promotes re-oxidation of the CF tip adjacent to the electrode as shown through simulation results in Fig. 4-15 for the medium density forming case from Fig. 4-14. The spatial distribution, achieved under the forming conditions determines the “supply” region, from which the O-ions might be
driven back to the CF during the reset operation. During reset, under the opposite voltage bias (vs. FORMING) oxygen ions accumulate near the bottom electrode (which must be a good oxygen diffusion barrier as to avoid the irreversible loss of oxygen during repeated switching cycles). The simulations show that when a sufficiently high density of O\(^{2-}\) ions accumulate next to the bottom electrode, they tend to diffuse towards the conductive filament edges which is enabled by a high local temperature and driven by a density gradient and mutual Coulomb repulsion. The filament works as an oxygen sink; oxygen ions readily bond to the available Hf atoms, thus, creating space for the next oxygen ion to move towards the filament.

Keeping the initial pinned vacancy distribution constant at ‘medium density’ as in Fig. 4-14c1, simulations exploring the effect of input pulse time and amplitude on RESET barrier formation are seen in Figs. 4-16(i) and 4-16(ii) respectively. It should again be noted that Figure 4-16 has been separated into two separate parts for better clarity. In both Figs. 4-16(i) and 4-12(ii) FORMING was carried out once with a forming voltage of 1.7 V and an I\(_{\text{COMPL}}\) of ~90 µA while RESET was subsequently done multiple times. In Fig. 4-16(i) RESET was done twice under two different pulse widths (500 and 100 ns) and pulse rise/fall times (2 ns and 10 ns) with a pulse amplitude held constant at -1.5 V. In Fig. 4-16(ii) RESET pulse width was held constant at 100 ns while pulse height varied (-0.6 V, -1.1 V and -1.6 V). These simulation results demonstrate that a longer reset pulse (as well as higher reset voltage) lead to re-oxidation of a larger section of the CF resulting in a less conductive HRS, consistent with the literature reports on the voltage vs. time trade-off [24, 82].

Fig. 4-16 A, A’, and A” presents 3D resistivity maps for the correspondingly marked moments in Fig. 4-16(i) for the 500 ns reset pulse. The 2D maps labeled Top, Mid-Low and Bottom correspond to the cross-sections in A, A’, and A” at the distances Top (z = 4 nm), Mid-Low (z = 0.5 nm), and Bottom (z = 2.5 Å) from the bottom electrode. The 3D maps of
Figure 4-16(PART1): Effect of \textit{RESET} pulse time on the barrier formation. (i) Simulated I-V characteristics for one device under pulse forming with $V_F$ amplitude of 1.7 V. Reset was done under 2 different pulse times. (ii) Effect of reset voltage (instead of pulse time) on HRS \textit{RESET} gap formation. Here, simulated I-V characteristics for \textit{RESET}s with 3 different pulse heights (i.e. voltages) are shown which were simulated following a similar forming process as in (i). The vertical portions of the I-V curves correspond to the resistance increase at a maximum pulse voltage during the pulse width.
4.4.3 Effect of Oxide Sub-Stoichiometry on RESET Gap Formation

In stoichiometric HfO$_2$, where the vacancy density of the GBs is relatively low (Fig. 4-14a), O ions are not prevented from out-diffusing away from the filament region which leads to an inability to \textit{RESET} the device as seen in the \textit{RESET} plot in Fig. 4-17 (green circles labeled no defects in the bulk) which corresponds to the simulated device described in Fig. 4-14a. With higher initial vacancy densities (as seen in Fig. 4-14b1, and 4-14c1) oxygen ions released during \textit{FORMING}, are better confined around the filament subsequently leading to the formation of a thicker dielectric barrier i.e. higher HRS resistance during \textit{RESET} as seen in Fig. 4-18. At a very high vacancy density, filament formation requires less additional vacancy generation (hence, less oxygen ion generation) in order to reach a compliance limit, which results in insufficient amount of available oxygen to form a barrier during \textit{RESET} (Fig. 4-17). Higher HRS resistance is obtained for the higher vacancy density cases (compare stoichiometric, low and medium). Both stoichiometric and high vacancy density cases exhibit poor \textit{RESET} due lack of oxygen ions in the filament vicinity and low
number of generated oxygen ions, respectively. In comparing these simulation results with experimental results of varying levels of dielectric sub-stoichiometry the same trends can be seen (See Chapter 2 Section 2.4.2; Fig. 2-20b)

Figure 4-17: Simulation results on the effect of initial vacancy density on RESET. Simulations were performed using the device structures in Fig. 4-14. The RESET pulse was 100 ns/-1.5 V. Both stoichiometric and high vacancy density cases exhibit poor RESET due lack of oxygen ions in the filament vicinity and low number of generated oxygen ions, respectively.
Figure 4-18: Simulation results on the effect of initial vacancy density on RESET. Simulations were performed using the device structures in Fig. 4-12. The RESET pulse was 100 ns/-1.5 V. Higher HRS resistance is obtained for the higher vacancy density cases (compare Fig. 4-17 with these low and medium densities).

Figure 4-19 compares our simulated (Fig. 4-14(ii) results) with experimentally measured RESET HRS resistances from devices with different levels of sub stoichiometry as determined by the OEL/HfO₂ ratio (see insert Table I).

Simulated RESET/SET cycling, shown in Fig. 4-20, compares reasonably well to the experimental pulsed switching indicating the capability of this model to describe essential physical features of the hafnia-based ReRAM switching process and LRS and HRS states. The set operation current is limited by the same current compliance level used during the forming event. Switching from a reset (high-resistance) to a set (low-resistive) state leads to the initial conductive filament being restored. For this to happen, the thin dielectric barrier
formed during reset needs to be broken. Due to the relatively high conductivity of the undisrupted portion of the ‘conductive’ filament, most of the applied voltage during set drops across the dielectric barrier (see Fig. 4-16A”). If the applied voltage (in particular, the pulse amplitude) is sufficiently high so that the electric field in this thin dielectric layer approaches the intrinsic dielectric strength of HfO₂, the Hf-O bond breakage proceeds very fast resulting in an equally fast release of the oxygen-ions and restoration of the conductive filament toward its post-forming shape. The description of the set process generally follows that of the forming process discussed above and the preliminary RESET/SET results in Fig. 4-20 demonstrate the validity of the prosed model.

Figure 4-19: The measured HRS resistance in HfO₂-based RRAM devices with different levels of sub stoichiometry as determined by the OEL/HfO₂ ratio (see insert Table I) after the DC RESET performed with different RESET voltages. Simulated values are taken from the data in Fig. 4-14 (ii).
4.4.4 Discussion

These simulation results support the initially-proposed physical model describing the conductive filament formation as having resulted from metal-oxygen bond breakage and subsequent outdiffusion of the released oxygen ions to promote a highly oxygen vacancy rich region to form conductive filament.

The subsequent RESET process is controlled by re-oxidation of the filament tip near the FORMING-cathode (RESET-anode) electrode. These simulations reveal critical factors controlling ReRAM FORMING (a balance between the voltage, current compliance, and stoichiometry) and the voltage-time trade-off in the pulse RESET.

Simulations also indicate a sufficient density of vacancies in sub-stoichiometric oxide can effectively confine oxygen ions in the vicinity of the filament which enables repeatable switching and higher HRS resistance while excessive vacancy density suppresses
The latter phenomenon results from a lack of a critical density of generated oxygen-ion/vacancy defects during the fast transient region of the FORMING process.

Gettering by an OEL preferentially generates vacancies in GBs leading to more efficient RESET. Larger grains in the dielectric (thus less GBs) may require longer/higher amplitude pulses for deeper RESET.

The proposed simulation model allows simultaneous optimization with respect to the device structure (composition/dimensions) and desired operation conditions enabling device optimization for low cost embedded non-volatile memory applications.

4.5 Summary of Modeling Results

The aforementioned processes of bond breakage and oxygen diffusion were implemented in a simulation program. These processes cannot be described in a closed analytical form because of their probabilistic nature and essentially non-local characteristics and, therefore, require the use of Monte Carlo simulations based of the process rate equations. At each iteration corresponding to a certain time interval, the program calculates the voltage across the RRAM taking into account the effective load resistor, and then current and corresponding 3D local electric field and temperature maps at each point in the simulated cell. Next, the probabilities of the vacancy generation (oxygen release) and oxygen hopping (for the already released oxygen) are calculated for each oxygen ion throughout the entire volume. After each vacancy generation or ion-vacancy recombination event, the change in stoichiometry is translated into a change in the local resistivity at the corresponding xyz location, from which a new field and temperature distribution is then recalculated.

The released oxygen ions, which remain in the filament’s immediate vicinity, tend to recombine with the vacancies there resulting in the final CF characteristics to depend, not
only on the current compliance value, but, also on a spatial distribution of the released oxygen ions by the end of the forming process. The post-forming ion distribution is the critical factor affecting the efficiency of the subsequent reset process and is controlled by the morphology (i.e. crystallinity) of the dielectric material. Indeed, due to a very low $O^{2-}$ hopping barrier (0.7 eV), the oxygen ion diffusion proceeds very effectively at room temperature which would lead to an eventual uniform distribution of expelled oxygen ions throughout the entire dielectric volume.

The proposed detailed explanation of the filament formation progression connects electrical characteristics of the dielectric to its morphology, which promotes an atomic-level description of this process. *The outcome of this process - an oxygen deficient region constituting a conductive filament surrounded by the interstitial oxygen ions expelled from the filament region – establishes the initial structural conditions for the subsequent RESET and SET operations.*
CHAPTER 5: Conclusions and Future Directions

5.1 Summary and Conclusions

In Chapter 2 preliminary electrical and physical characterization work outlined the importance of reducing both forming voltage and overshoot-current in filament-based TMOs. It also provided experimental proof of the links between sub-stoichiometry (MeOₓ/OEL thickness ratios) and stable RESET/SET switching. Sub-stoichiometry was then defined experimentally as a key factor in reversible filament switching: Suggesting a defect-assisted dominated mechanism for HfO₂-based ReRAM. ReRAM filament ‘growth’ was then aligned with dielectric breakdown theory and highlighted as having key mechanisms of bond breakage and oxygen outdiffusion.

Filament-formation (or ‘growth’) was experimentally characterized in Chapter 3 by defining the effects of current, voltage, and ambient temperature on LRS and HRS stability and endurance. It was clearly evident that local filament temperature and forming voltage (Vᵢ) can be used to control the filament size and stability, respectively. A ‘hot’ FORMING method was also proposed and evaluated which improved device-to-device and cycle-to-cycle uniformity of the LRS and HRS resistances, as well as increased corresponding device memory windows.

Understanding how and why ‘hot’-FORMING works on a microscopic level was the primary focus of Chapter 4. An investigation was presented regarding the factors controlling the conductive filament stability using a Monte-Carlo physics-based model to simulate the thermal runaway phase of the FORMING process. The simulations, which were in alignment with the presented theory in Chapter 2, suggested filament growth during this second stage of FORMING is due to a local field increase which is sufficiently high as to lower
the effective activation energy for vacancy generation below that of the oxygen diffusion and recombination processes. As a consequence, a significant number of recombination events take place during the post-\textit{FORMING}-relaxation phase leading to the formation of a less robust and less stable filament after high voltage \textit{FORMING} compared with low voltage \textit{FORMING} conditions for both 25°C and 150 °C. 'Hot' forming was simulated; and shown to increase the size of the conductive-filament cross-sections as well as lower the $V_F$. Both of which serve in producing a better understanding of '\textit{why}' and '\textit{how}' increasing the ambient temperature by just a 100°C establishes better control over low-power switching variability. This was all proven while staying consistent with the experimental results presented in Chapter 3.

Chapter 4 further investigated the role of sub-stoichiometry in terms linking it with the enablement of \textit{RESET/SET} switching. Simulations, which matched experimental data, promoted an atomic-level description of the oxygen-ion-distributions after-\textit{FORMING} as a function of increased-numbers-of-pinned-vacancies with higher levels of dielectric sub-stoichiometry. \textit{The outcome of this process - an Hf-metal rich conductive filament surrounded by the interstitial oxygen ions expelled from the filament region - establishes the initial structural conditions for the subsequent \textit{RESET} and \textit{SET} operations.} A dielectric theory based simulation framework has been developed for the better enablement of linking ReRAM operational mechanisms to materials properties such as dielectric stoichiometry, OEL metal layer oxygen-affinity, and different types of electrodes.

This dissertation opens the path to ReRAM performance optimization by (1) providing key fundamental guidelines for \textit{‘ReRAM defect-engineering’} and by (2) developing a simulation tool that can be used to further investigate the naturally-stochastic ionic-based ReRAM device structure. It is the author's hope that this work is further developed and used in expediting low-cost embedded (NVM) ReRAM applications.
5.2 Suggested Future Directions

a. Suggestion 1: Merge ab-initio molecular calculations and trap assisted tunneling results with these thermal runaway (effective resistance model) simulations as to create a complete simulation model. See Fig. 5-1 and Fig. 5-2. Note that the effective resistance model used in this work underestimates the amount of ions that reach the Ti oxygen exchange layer (OEL) during the initial leakage phase of the forming process. Nevertheless, this doesn’t affect the results discussed in the following, which are related only to the O-2/V distributions inside the HfOx layer.

b. Suggestion 2: SET/RESET switching endurance simulations. Preliminary MATLAB code has already been developed by the author.

c. Suggestions 3: Retention simulations. Preliminary MATLAB code has already been developed by the author.

Figure 5-1: Comparing simulations of T.A.T. with this model’s drift-diffusion simulation. All data is simulated.
Simulation Correct Trend $V_F$ with Ambient Temp. However, Need T.A.T. for completion

![I-V curve Simulation (Effective Resistance Model) - no TAT](image)

Ion diffusion speed increases with ambient temp. → ‘Cleaning out’ CF region → Reducing amount recombination after $I_{comp}$ is meet → stabilizing CF

This is all related to $V_F$ and the $E_{field}$

$V_F$ is different at each temp. → Changes both $E_{field}$ and Temp., which effect Gen/Recomb./Diffusion differently

Temp. 425K → higher temp. combined with lower $E_{field}$
Temp. 77K → higher $E_{field}$ combined with lower temp.

Figure 5-2: Shows simulation results with 3D ion/vac distributions. This figure also gives an explanation of how the ions diffuse with increased temperature. I-V curve (simulations) needs to be updated (as explained in Fig. 5-1).
APPENDIX 1: Other NVM Emerging Technologies

(PCM, STT-MRAM, CNT-RAM)

Overview of Phase Change Memory is shown in Fig. APEX1-1. Fig. APEX1-1a shows the mechanism of changing from crystalline (SET=1) to Amorphous (RESET=0). Figure APEX1-1b shows the typical PCM IV curves while section (c) shows a SET to RESET Transitions – changing from crystalline to amorphous. Figure APEX1-1d shows the RESET to SET transitions. This figure shows the typical current vs. time relations next to electro-thermal simulations of the transformation from amorphous back-into the crystalline phase.

Figure APEX1-1: Overview of Phase Change Memory. A) Mechanism of changing from crystalline (SET=1) to Amorphous (RESET=0). B) Typical IV curve C) SET to RESET Transitions – changing from crystalline to amorphous. D) RESET to SET transitions showing typical current vs. time relations next to the electro-thermal simulations of the transformation from amorphous back-into the crystalline phase.
Figure APEX1-2 shows an overview of magnetic memory. This figure summarizes the fundamentals of magnetic memory in terms of basic Spintronics, magneto-resistance how information is stored inside the free layers of the device (see Fig. APEX1-2a). In Fig. APEX1-2b the difference between Toggle (E-field driven) and Spin-Transfer Torque MRAM is defined. Fig. APEX1-2b also describes the pros and cons of MRAM technology.

Figure APEX1-3 describes the fundamental different types of carbon nanotubes in terms of chirality. There are two main types of carbon-nanotube-based memory the first type uses the various transistors like electrical properties of the nanotubes in terms of moving a bucky-ball (spherical CNT structure) around with electric field and currents.
Issues with this type of memory included making the terminal connections and other manufacturing uses. The second type is currently being manufactured by a company named Nantero (http://www.nantero.com/index.html) This type of CNT-based memory uses the following two principles about CNTS and is illustrated in Figure APEX1-4a through d:

1) Metallic nanotubes will bend toward a perpendicular semiconducting nanotube when electrically charged.

2) When a metallic nanotube is one to two nanometers away from a semiconducting nanotube, the electrical resistance at the junction is low, creating an ON state. When the nanotubes are apart the resistance is much higher, creating an OFF state.

Figure APEX1-3 Fundamentals of carbon nanotubes in terms of different types.
Figure APEX1-4 (a) Diagram labeling the many parts of the CNT-based memory. (b) The applied voltage bends the CNTs into the ON position. (c-d) shows the respective mechanical strain, van der Waals energy and total energy plots that can be used to explain the OFF and ON state of the device. Source: http://www.nantero.com/index.html.
APPENDIX 2: Integrated 1T1R – Processing Steps

**Gate litho**
- Target CD 100nm
- Tri-layer litho

**Gate etch**
- Etch thru high k
- Stop on thermal gate oxide
Post gate RIE wet etch (Ox)

1T:1R NLDD/NHALO
- (50A SiN) Seal

P Well ~ 1E17 B doped
1T1R NLDD/NHALO
- (50A SiN) Seal
- As (N+)

1T1R NLDD/NHALO
- (50A SiN) Seal
- As (N+)
- Halo P (N++) implant to prevent punch-through
- (<100A thickness) SDE: N++
  High Dopant Density
Integrated 11Tr – Process Integration

RIE etch of Spacer

Integrated 11Tr – Process Integration

As S/D implant (drive in step)
Integrated 1T1r – Process Integration

SD resist strip

Integrated 1T1r – Process Integration

Spike anneal
(electrical activation step)

Spike anneal in x% O2 in N2
Integrated 1T1r - Process Integration

Nickel Deposition

Anneal → Nickel Silicide
Integrated 11r – Process Integration

Nickel Removal

NiSi Anneal
Forming gas anneal

Passivation Nitride

HARP
Integrated 111r - Process Integration

CMP polish
USG Dep.

sacrificial oxide (Dr. USG)

Integrated 111r - Process Integration

CA litho
- Tri-layer litho

CA Etch
- Wet Clean SC1

(Cleaning) SC1

TiN linear (Ti/TiN)
Integrated 1T1 – Process Integration

Fill Trench with B-doped a-Si
RTP (activation of B)

B-doped a-Si

Integrated 1T1 – Process Integration

Wet Clean

CA Poly Polish
Wet Clean: NH4OH
Figure APEX 2-1: Summary of Process Integration for integrated 1T1R testing structure. Two ReRAM crossbar devices (50x50nm² and greater in size) was fabricated on-top of the metal-1 layer of the transistor. This processing was done in the CNSE 300mm fab under the guidance from the SEAMTECH process integration team: David Gilmer, Bill Taylor, MinGyu Sung, Sergei Koveshnikov (Intel assignee), Kristina-Young Fisher (Global-Foundries Assignee), Shweta Deora and Brian Butcher (SEAMTECH Intern/CNSE graduate student).
Figure APEX2-2: Larger view of Fig. APEX 2-1
APPENDIX 3: Description of Simulating 3D Grains/Grain-Boundaries

This Appendix steps through the process of making a 3D grain/grain boundary. Code for this process is in Appendix 5.
Figure APEX 3-1: Varying the starting points of each pinned vacancy.
Figure APEX 3-2: Varying the directions of the smooth grain walls

Could study the effect of grain size on ReRAM variability.
Figure 3-3: varying the directions and thickness of the grain boundary walls.
Figure APEX 3-4: Varying the directions of the grain boundary walls.

Figure APEX 3-5: Varying directions of ‘non-smooth walls’ with random deletions of 50% of $V+2$ pinned.

Varying directions of ‘non-smooth walls’ with random deletions of 10% of $V+2$ pinned.

Rand 10% to 50% pre radius

Figure APEX 3-5: Varying directions of non-smooth walls with the random deletions of 5-% of $V+2$ pinned.
APPENDIX 4: Executing Numerical Calculations in MATLAB

This Appendix (using the three figures below) explains in more detail how the numerical method of solving the potential and temperature were done in the MATLAB code. The actually code is found in Appendix 5. Figure APEX 4-1 shows a more detailed flow chart.

![Program Description (3D Map creation)](image)

Figure APEX 4-1: A more detailed version of the program flow diagram.

(1T1R) R\text{\textit{\scriptsize{Channel}}} in series with MIM stack: TiN / Ti / HfO\text{\textsubscript{2}} / TiN:

- Material Properties defined w/ thermal conductance and resistivity
- Initial Vac. distribution
  - 0.5nm radius*, random #vac. in bulk (not shown here)
  - Higher concentration closer to Ti layer

**Main loop:** numerically solving Fourier heat-transfer, charge continuity equations in ea. XYZ bin

$$ \nabla \cdot \mathbf{\sigma} \nabla V = 0 $$

$$ -\nabla \cdot k_{ij} \nabla T = \mathbf{\sigma} \nabla V \right| $$

- **Resistivity**— local values depending on # of vacancies (stoichiometry)**
- **Potential**
- **Power dissipation:**
- **Temp. / E-Field**

Event(s) $$\rightarrow$$ 3D Map Update
**SOR (Successive Over Relaxation) Numerical Method → Solving Potential, Temp., in a convergence loop**

At this point a matrix of 3-D maps has been created in terms of $V_{ReRAM}$, thermal conductance, and $\rho_o(ReRAM)$.

Finds ~solution for V and T using 3D Fourier/ Poisson equations in the matrix form $Ax=b$:

$$A = 
\begin{bmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}, \quad x = 
\begin{bmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_n
\end{bmatrix}, \quad b = 
\begin{bmatrix}
  b_1 \\
  b_2 \\
  \vdots \\
  b_n
\end{bmatrix}.$$

Splitting $A$ into: $A=D+L+U$

$$(D + \omega L)x = \omega b - [\omega U + (\omega - 1) D]x$$

Using a relaxation factor $\omega \rightarrow$ speeds up convergence of a slow-converging process.

---

**Figure APEX 4-2: Matrix setup**

**SOR (Successive Over Relaxation) Numerical Method → Solving Potential, Temp., in a convergence loop**

Finds ~solution for V and T using 3D Fourier/ Poisson equations in the form $Ax=b$:

$$A(i,j,k)^* x(i,j,k)-x_0 = P(i,j,k)$$

In case of potential (Poisson equation)

$x = \psi(i,j,k)$ is the potential

$b = Q(i,j,k)$ the charge density ($\text{C/m}^3$),

$A = \text{Electrical capacitance (F)}$

Temperature (Fourier stationary equation)

$x = T(i,j,k)$

$b = P(i,j,k)$ power density ($\text{W/m}^3$)

$A = \text{Inverse thermal resistances (mK/W)}$

BC ($z$) : (Ambient Temp: 0, V)

BC ($x,y$): Lateral heat exchange neglected

---

**Figure APEX 4-3: Parameters used for each respective loop.**
APPENDIX 5: Matlab Code

In this appendix is pasted the MATLAB code that was created in this dissertation. It is organized by .m file. It was developed by the author in collaboration with Luca Vandelli (Vandalo) from the University of Modena and Reggio Emilia. Specific functions compiled in C code (described in Appendix 4) for both Windows and Linux platforms were embedded into the MATALB code below.

```matlab
% Change default axes fonts.
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 14)

% Change default text fonts.
set(0,'DefaultTextFontname', 'Times New Roman')
```
set(0,'DefaultTextFontSize', 14);
set(0,'fontWeight', 'bold')

% - Start of program
% - Clears all variables in workspace, closes all matlab windows
% - Turns on warning
% - Option to turn on profiler, if on at end of program will be able to
determine where you program is the slowest, if off program will run
% faster
keyboard;
clear all; close all;
warning on all;
profile off;

% - Creates fresh storage folder for output files (binary files stored)
if exist('Output','dir')
    rmdir('Output','s');
end
mkdir('Output');

% - Tracks total time program is running
tic;
telapsed = toc(tic); %options = optimset('UseParallel','always');

% - Device initialization
startup; % links all folders
global layers; % empty struct
global X1_D X2_D Y1_D Y2_D;

InitializeDevice(); % fills struct with layer parameters read from txt file
% also initializes boundary conditions, used in other
% functions
% - At this point layer parameters have been created --> only layers.t is
% used, which defines the upper bound of a layer
% - Next step is to link each layer with material properties of interest

% - Resistivity of each material defined
roHfO2=1e5; % Ohm - meters
roHf=4e-5;
roTi=1/7.407e5;
roTiN=1/3.3e6;

% - Thermal conductance of each material defined
kthHfO2=0.5; % Watts/mKelvin
kthHf=23;

% - Resistance for TE and BE set very very low (metal), for use in SOR func
Rinf=1e-100; % Ohms
Rsup=1e-100;
Rthinf=1e-100;
Rthsup=1e-100;

% - Creates a grid in 3D space, sectioned into cubic bins (2.5A^3)
Nx=floor(61*1);
Ny=floor(81*1);
Nz=floor(81*1);
xspacing=linspace(-10e-9,10e-9,Nx); % - Creates sections
yspacing=linspace(-10e-9,10e-9,Ny);
zspacing=linspace(-2e-9,10e-9,Nz); %BE is a barrier, no need to grid it
[xmesh,ymesh,zmesh]=meshgrid(xspacing,yspacing,zspacing); % - Creates grid
% - Definition of the initial structure
% - Links resistivity and thermal resistance with each cubic bin
% - NEED TO UPDATE LAYER USAGE HERE (non consistent)
% - This is done by assigning the borders of each cubic bin a value
% - Assigning all x direction lines

Xc=(xspacing(1:end-1)+xspacing(2:end))/2;
[X1 Y1 Z1]=meshgrid(Xc,yspacing,zspacing);
rox=roHfO2*ones(size(X1));   % initializes every bin with HfO2
kthx=kthHfO2*ones(size(X1));
rox(Z1<=0)=roTiN; % changes BE to TiN
kthx(Z1<=0)=kthTiN;
rox(Z1>5e-9&Z1<=8e-9)=roTi; % changes OEL to Ti
kthx(Z1>5e-9&Z1<=8e-9)=kthTi;
rox(Z1>8e-9&Z1<=10e-9)=roTiN; % changes TE to TiN
kthx(Z1>8e-9&Z1<=10e-9)=kthTiN;

% - Assigning all y direction lines

Yc=(yspacing(1:end-1)+yspacing(2:end))/2;
[X2 Y2 Z2]=meshgrid(xspacing,Yc,zspacing);
roy=roHfO2*ones(size(X2));
kthy=kthHfO2*ones(size(X2));
roy(Z2<=0)=roTiN;
kthy(Z2<=0)=kthTiN;
roy(Z2>5e-9&Z2<=8e-9)=roTi;
kthy(Z2>5e-9&Z2<=8e-9)=kthTi;
roy(Z2>8e-9&Z2<=10e-9)=roTiN;
kthy(Z2>8e-9&Z2<=10e-9)=kthTiN;

% - Assigning all z direction lines

Zc=(zspacing(1:end-1)+zspacing(2:end))/2;
[X3 Y3 Z3]=meshgrid(xspacing,yspacing,Zc);
roz=roHfO2*ones(size(X3));
kthz=kthHfO2*ones(size(X3));
roz(Z3<=0)=roTiN;
kthz(Z3<=0)=kthTiN;
roz(Z3>5e-9&Z3<=8e-9)=roTi;
kthz(Z3>5e-9&Z3<=8e-9)=kthTi;
roz(Z3>8e-9&Z3<=10e-9)=roTiN;
kthz(Z3>8e-9&Z3<=10e-9)=kthTiN;

%- Initial grid of Oxygen Ion postions created

%--------------------------------------------------------------------------

% ArcLengthx=5e-10;
% r_pinned=2.5e-10;
% theta_grain=1.5; %120deg
% theta_variation=ArcLengthx/r_pinned;
% max_zspacing=1e-9;
% precent_to_delete=.94;
% precent_to_delete2=.92;
% flag=1;
% [PinnedVacMap, rad_pos]=grainboundries(theta_grain,
theta_variation,zspacing,precent_to_delete,precent_to_delete2, r_pinned, flag,
1);

flag=0;
while r_pinned<=10.5e-9
 r_pinned=r_pinned+.25e-9;
 theta_variation=ArcLengthx/r_pinned;
 [temp, rad_pos]=grainboundries(theta_grain,
theta_variation,zspacing,precent_to_delete,precent_to_delete2, r_pinned, flag,
rad_pos);

168
% PinnedVacMap=vertcat(PinnedVacMap, temp);
%
end

Cir=[];

% ArcLengthx=ArcLengthx/2;
% r_pinned=6e-9;
% theta_grain=.25; %120deg
% precent_to_delete=.94;
% precent_to_delete2=.91;

flag=2;
jj=2.55e-9; % moves out edges of cir to make it fit square better
jjj=0;
while r_pinned<=6.25e-9
    r_pinned=r_pinned+.25e-9;
    theta_variation=ArcLengthx/r_pinned;
    [Cirtemp, rad_pos]=grainboundries(theta_grain,
        theta_variation, zspacing, precent_to_delete, precent_to_delete2, r_pinned, flag,
        rad_pos, jj, jjj);
    Cir=vertcat(Cir, Cirtemp);
end

VacMap2=vertcat(PinnedVacMap, Cir);

IonMap=zeros(0,3);

%------------------------------------------------------------------------
--

[thetha, rho, zz]=cart2pol(VacMap2(:,1), VacMap2(:,2), VacMap2(:,3));
polArray=[thetha, rho, zz];
polArray(polArray(:,2)>.5e-9)=0;

for i=1:length(polArray(:,1))
    if polArray(i,1)==0;
        polArray(i,2:3)=0;
        count=count+1;
    end
end

mSize=i-count;
polArray(polArray==0)=[];
polArray=reshape(polArray, mSize, []);

[VacMap0x VacMap0y] = pol2cart(polArray(:,1), polArray(:,2), polArray(:,3));
VacMap0=[VacMap0x, VacMap0y, VacMap0z];

VacMap2=roundn(VacMap2, -14);
VacMap0=roundn(VacMap0, -14);
for i=1:length(VacMap0(:,1))
    for j=1:length(VacMap2(:,1))
        if VacMap2(j,1:3)==VacMap0(i,1:3);
            VacMap2(j,1:3)=Inf;
        end
    end
end

fid=fopen(['FVacMap2_n', num2str(1)], 'rb');
VacMapxx=reshape(fread(fid, inf, 'double'), [], 3);
fclose(fid);
VacMap2=VacMapxx;

fid=fopen(['FVacMap0_n', num2str(1)], 'rb');
VacMapxx=reshape(fread(fid,inf,'double'),[],3);
fclose(fid);
VacMap0=VacMapxx;

IonMap=[.5e-10 .5e-10 5.5e-9];
VacMap0(1,:)=[];
VacMap0(2,:)=[];
VacMap0(4,3)=VacMap0(4,3)+.78e-9;
VacMap0(5,3)=VacMap0(5,3)+.75e-9;
% VacMap0(2,3)=VacMap0(2,3)+3e-9;
% VacMap0(6,3)=VacMap0(5,3)+.75e-10;
% VacMap0(7,3)=VacMap0(2,3)+3e-10;
% VacMap0(8,2)=VacMap0(2,2)+6e-10;
keyboard;% ----- Voltage Sweep RESET Mode (1001 step, Forming)---------------------------%
PULSE_OFF=1;
Tr_Tf_TimeTotal=10e-9;  %timesec
FormPW=500e-9;          %timesec
FormPH=1.75;              %V
%FormPW=100e-9;       %timesec
%FormPH=1.3 ;           %V
ResetPW=100e-9;
ResetPH=-1.75;
SetPW=100e-9;
SetPH=1.1;
RelaxationTime=1.5e-6;

[~,~,~,~,~,~,~,~, Vin,tin, TotTimeForm, TotTimeRelaxReset, TotTimeRelaxSet1,
TotTimeRelaxSet2, TotTimeRelaxSet3, TotTimeRelaxSet4, TotTimeRelaxSet5,
TotTimeRelaxSet6, TotTimeRelaxSet7, TotTimeRelaxSet8, TotTimeRelaxSet9,
TotTimeRelaxSet10, TotTimeRelaxSet11, TotTimeRelaxSet11
TotTimeRelaxSet12]=PulseTrain(Tr_Tf_TimeTotal,FormPW,FormPH,ResetPW,ResetPH,SetPW,SetPH,RelaxationTime);
figure;
plot(tin,Vin);
xlim([-5e-7 tin(end)+5e-7]);
nVin=1;
Vtotal=Vin(nVin);
Icompl=50e-6;
keyboard;

Rload=1e-1;  %20uA

% - Temperature coeffient
alpha_T=0.0037;
% - Ambient Temperature
%Te=423.15;
%Te=398.1
Te=300;

% ----- Pulsed Forming, assuming no rise, fall time, only pulse width------%
PULSE_OFF=0;
% Vd=2;
% Vin = linspace(Vd,Vd,1);
% vramp_rate=.00000001;  % V/s
% tin = abs(Vin/vramp_rate);
% nVin=0;
% Vtotal=Vin(nVin+1);
% %-----------------------------------------------%
% - Make initial current = 0, parasitic capacitance value, flag (compl) which
% is used to signal when to apply t_after_Icompl
I=0;
Cp=1e-15;
Cp=1e-21;
compl=0;
tau=0;
Vss=Vtotal;
Rreram=Inf;

% --------------------------------- Initialize the loop --------------------------%
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%

nstep=1;
t=[]; %time
tottime=0;
VPAt=vpa(t,40); % time with digits below 1e-15
tsim_max_atIcompl=vpa(1,40);
VPAtottime=vpa(0,40);
nstepmax=1e10; % no stop until after Icompl is reached
tsim_max_withV_onReRAM=1e10;
tsim_max=1000000000000000000000000; % sec; 1,157 days
stop =0;
% used for recomb and spacer Ion tracking
rec =[]; tSpacer=[];
rec_pos=[]; Spacer_pos=[];
nrec=1; nSpacer=1;
ngen=0;
ngen_count=0;
V=zeros(length(yspacing),length(xspacing),length(zspacing)); % potential
T=Te*ones(length(yspacing),length(xspacing),length(zspacing)); % temp
P_maps=[]; % Power
T_maps=[];
V_maps=[];
IonsInTE=[];
IonsInMeOx=[];
VssReRAM=Vtotal;
CountIonsInTE=0;
CountIonsInMeOx=0;
nstepstopx=1;
ngen_count=0;
FlagAll=1;       % turns on Diff Gen and Recomb if 1
cheat_5e_neg6=0; % used to save data, in case need to stop prog
cheat_9e_neg6=0;
cheat_19e_neg6=0;
cheat_29e_neg6=0;
cheat_39e_neg6=0;
cheat_49e_neg6=0;
cheat_55e_neg6=0;
cheat_99e_neg6=0;
onetime=1;
VoltageON=1;
flag=0;
SOR_on=1; % SOR on
lifetime=1e45;
restriction=1;
IcomplMeet=0;
SpacerIonMap=[];
Aft_time=0;
InCF_Precentages=[100 100];
Rcutoff=[Inf 1e12 5e11 1e11 5e10 2.5e10 1e10 5e9 2.5e9 1e9 5e8 2.5e8 1e8
5e7 2.5e7 1e7 5e6 2.5e6 1e6 5e5 2.5e5 1e5 5e4 2.5e4 1e4 5e3 2.5e3 1e3 5e2 2.5e2
1e2 5e1 2.5e1 1e1 5e0 2.5e0 1e0];
R_now=1;
sexflag=0;
rCF_array=[0 .25e-9];
rCF=.25e-9;
ReverseBkSweep=0;
VpinChargeFlag=2;
NoRecomb=NaN;
RecombMap=[];
TurnOFFGEN_Recomb_STEP=0;
ResetFlag=0; % if 1 then reset conditions apply
FormingFlag=0;
SetFlag=0;
% _______________________________________________________
% _______________________________________________________
% _______________________________________________________
while ~stop
  % Update voltage applied to the RRAM device accounting for the delay
  % associated to the parasitic capacitance
  if tottime<=TotTimeForm
    FormingFlag=1;
    ResetFlag=0; % if 1 then reset conditions apply
    SetFlag=0;
  end
  if (tottime>TotTimeForm && tottime<=TotTimeRelaxReset) ||
    (tottime>TotTimeRelaxSet1 & & tottime<TotTimeRelaxReset2) ||
    (tottime>TotTimeRelaxSet2 & & tottime<TotTimeRelaxReset3) ||
    (tottime>TotTimeRelaxSet3 & & tottime<TotTimeRelaxReset4) ||
    (tottime>TotTimeRelaxSet4 & & tottime<TotTimeRelaxReset5) ||
    (tottime>TotTimeRelaxSet5 & & tottime<TotTimeRelaxReset6) ||
    (tottime>TotTimeRelaxSet6 & & tottime<TotTimeRelaxReset7) ||
    (tottime>TotTimeRelaxSet7 & & tottime<TotTimeRelaxReset8) ||
    (tottime>TotTimeRelaxSet8 & & tottime<TotTimeRelaxReset9) ||
    (tottime>TotTimeRelaxSet9 & & tottime<TotTimeRelaxReset10) ||
    (tottime>TotTimeRelaxSet10 & & tottime<TotTimeRelaxReset11) ||
    (tottime>TotTimeRelaxSet11 & & tottime<TotTimeRelaxReset12);
    FormingFlag=0;
    ResetFlag=1; % if 1 then reset conditions apply
    SetFlag=0;
    Rload=0;
    compl=0; % reset the compliance flag so subsequent set works ok
  end
  if (tottime>TotTimeRelaxReset & & tottime<TotTimeRelaxSet1) ||
    (tottime>TotTimeRelaxReset2 & & tottime<TotTimeRelaxSet2) ||
    (tottime>TotTimeRelaxReset3 & & tottime<TotTimeRelaxSet3) ||
    (tottime>TotTimeRelaxReset4 & & tottime<TotTimeRelaxSet4) ||
    (tottime>TotTimeRelaxReset5 & & tottime<TotTimeRelaxSet5) ||
    (tottime>TotTimeRelaxReset6 & & tottime<TotTimeRelaxSet6) ||
    (tottime>TotTimeRelaxReset7 & & tottime<TotTimeRelaxSet7) ||
    (tottime>TotTimeRelaxReset8 & & tottime<TotTimeRelaxSet8) ||
    (tottime>TotTimeRelaxReset9 & & tottime<TotTimeRelaxSet9) ||
    (tottime>TotTimeRelaxReset10 & & tottime<TotTimeRelaxSet10) ||
    (tottime>TotTimeRelaxReset11 & & tottime<TotTimeRelaxSet11);
    FormingFlag=0;
    ResetFlag=0; % if 1 then reset conditions apply
    SetFlag=1;
    % keyboard;
  end
  if Icompl==50e-6
    MinChannelRes=Vtotal/50e-6; % cannot give more V to ReRAM, in terms of V
    division
  elseif Icompl==20e-6
    MinChannelRes=Vtotal/20e-6; % cannot give more V to ReRAM, in terms of V
    division (Max V on ReRAM)
  end
elseif Icompl==100e-6
    MinChannelRes=Vtotal/100e-6; %cannot give more V to ReRAM, in terms of V division (Max V on ReRAM)
elseif Icompl==75e-6
    MinChannelRes=Vtotal/75e-6;
end

if nstep >= 2 && Rreram<Inf
    if Vtotal==0; Rload=0; end;
    if Vtotal>0 && (compl==1) && ResetFlag==0 %Compl is meet
        % Rload=10;
        % if Icompl==20e-6; Rload=4.64e3; end;
        % if Icompl==100e-6; Rload=1e3; end;
        Rload=Rreram+10000; %all Voltage is split between Rload and RRAM
        Rload=10;
    end
    if Vtotal>0 && (compl==0) && ResetFlag==0% Compl not meet
        if Icompl==20e-6; Rload=4.64e3; end; %all voltage is on reram
        if Icompl==100e-6; Rload=1e3; end;
        if Icompl==50e-6; Rload=1.5e3; end;
        if Icompl==75e-6; Rload=10; end;
    end;
    if ResetFlag==1 %meaning Ream is controlling Current max
        Rload=0;
    end
end

tau=Cp*Rload*Rreram/(Rload+Rreram);
VssLoadR=Vtotal*(Rload/(Rload+Rreram));
VssReRAM=Vtotal-VssLoadR;

Current Compliance → Cal. with R_load in Series with ReRAM

\[
V_{reram} = V_{ss} + (V_{reram} - V_{ss})e^{-\frac{t-t_0}{\tau}}
\]

\[
tau = C_p R_{load} R_{reram} / (R_{load} + R_{reram})
\]

\[
V_{ssReRAM} = V_{total} - R_{load} \cdot t
\]

\[
V_{ssLoadR} = R_{load} \cdot t
\]
if \( V_{\text{ssReRAM}} < (V_{\text{total}} - R_{\text{load}} \cdot I_{\text{compl}}) \); \( V_{\text{ssReRAM}} = (V_{\text{total}} - R_{\text{load}} \cdot I_{\text{compl}}) \); end

if nstep == 2 || nstep == 3
    deltat = t(nstep-1);
else
    deltat = VPAt(end-1) - VPAt(end-2);
end
if deltat > 0
    if Rload ~= 0
        \V_{\text{reram}} = V_{\text{ssReRAM}} + (V_{\text{reram}} - V_{\text{ssReRAM}}) \cdot \exp(-\text{deltat} / \tau);
        \V_{\text{load}} = V_{\text{total}} - \V_{\text{reram}};
        \V_{\text{reram}} = \text{double} \left( \V_{\text{reram}} \right);
        \V_{\text{load}} = \text{double} \left( \V_{\text{load}} \right);
        \V_{\text{ssReRAM}} = \text{double} \left( \V_{\text{ssReRAM}} \right);
    else
        \V_{\text{reram}} = \V_{\text{ssReRAM}};
        \V_{\text{load}} = V_{\text{ssLoadR}};
    end
end
if deltat == 0
    warning('!!! deltat = 0!!! Time is not being recorded correctly');
end
else
% initializion steps,
\V_{\text{reram}} = \V_{\text{total}};
\V_{\text{load}} = 0.0000000001;
\V_{\text{ssLoadR}} = 0.00000001;
End
%--------------------------Used for plotting -----------------------------%
figure;
% if ~isempty(VacMap0)
% plot3(reshape(VacMap0(:,1),1,[])*1e9,reshape(VacMap0(:,2),1,[])*1e9,reshape(VacMap0(:,3),1,[])*1e9,'o','MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize ',7);
% hold on;
% if ~isempty(VacMap2)
% plot3(reshape(VacMap2(:,1),1,[])*1e9,reshape(VacMap2(:,2),1,[])*1e9,reshape(VacMap2(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor','k','MarkerSize ',4);
% hold on;
% end
if ~isempty(IonMap)
    plot3(reshape(IonMap(:,1),1,[])*1e9,reshape(IonMap(:,2),1,[])*1e9,reshape(IonMap(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',6);
    hold on;
end
if ~isempty(SpacerIonMap)
    plot3(reshape(SpacerIonMap(:,1),1,[])*1e9,reshape(SpacerIonMap(:,2),1,[])*1e9,reshape(SpacerIonMap(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor','cyan','MarkerSize',6);
    hold on;
end
Plot_layers(((X2_D-X1_D)*1e9)-(0),((Y2_D-Y1_D)*1e9)-(0),((layers(1).t)*1e9)),((layers(2).t)*1e9)),'',' ',' ',' ',' ','[]',.05,.5)
hold on;
keyboard;
pause(.5);
figure;
if ~isempty(VacMap0)
    plot3(reshape(VacMap0(:,1),1,[])*1e9,reshape(VacMap0(:,2),1,[])*1e9,reshape(VacMap0(:,3),1,[])*1e9,'o','MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize',7);
    hold on;
end
if ~isempty(VacMap2)
    plot3(reshape(VacMap2(:,1),1,[])*1e9,reshape(VacMap2(:,2),1,[])*1e9,reshape(VacMap2(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor','k','MarkerSize',4);
    hold on;
end
if ~isempty(IonMap)
    plot3(reshape(IonMap(:,1),1,[])*1e9,reshape(IonMap(:,2),1,[])*1e9,reshape(IonMap(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',6);
    hold on;
end
if ~isempty(SpacerIonMap)
    plot3(reshape(SpacerIonMap(:,1),1,[])*1e9,reshape(SpacerIonMap(:,2),1,[])*1e9,reshape(SpacerIonMap(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor','cyan','MarkerSize',6);
    hold on;
end
Plot_layers(((X2_D-X1_D)*1e9)-(0),((Y2_D-Y1_D)*1e9)-(0),((layers(1).t)*1e9)),((layers(2).t)*1e9)),'',' ',' ',' ',' ','[]',0.00,0.00)
view(0,90);
hold on;
keyboard;
pause(.1);
figure;
plot(zspacing,abs([squeeze(T(floor(length(xspacing)/2),floor(length(yspacing)/2),:)),squeeze(T(10,10,:))]);
    hold on;
% legend('Inside CF','Outside CF');
% title('T vs z');
figure;

plot3Dmap(xspacing,yspacing,zspacing(1,11:51),V,'PotentialMap');
hold off;
pause(.1);

 [~,~,Ezmesh]=gradient(V,xspacing(2)-xspacing(1),yspacing(2)-
yspacing(1),zspacing(2)-zspacing(1));
plot3Dmap(xspacing,yspacing,zspacing(1,11:51),abs(Ezmesh),'E-Field
(V/m)',-10e-9,10e-9);

 if nstep>2
 figure;
plot(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(:,15),T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(:,12));
title('I vs V_25C');
 pause(.1);
end

-----------------------------------------------
% - Using the 3D VacMap , Calculates defect density map (cm^-3)
xdens1=zeros(size(X1)); xdens2=zeros(size(X2)); xdens3=zeros(size(X3));

 xdens1(Z1>0 & Z1<=layers(1).t)=nmap(VacMap0,X1(Z1>0&Z1<=
layers(1).t),Y1(Z1>0&Z1<=
layers(1).t),Z1(Z1>0&Z1<=
layers(1).t),xspacing,yspacing,zspacing);

 xdens2(Z2>0 & Z2<=layers(1).t)=nmap(VacMap0,X2(Z2>0&Z2<=
layers(1).t),Y2(Z2>0&Z2<=
layers(1).t),Z2(Z2>0&Z2<=
layers(1).t),xspacing,yspacing,zspacing);

 xdens3(Z3>0 & Z3<=layers(1).t)=nmap(VacMap0,X3(Z3>0&Z3<=
layers(1).t),Y3(Z3>0&Z3<=
layers(1).t),Z3(Z3>0&Z3<=
layers(1).t),xspacing,yspacing,zspacing);

 rox(Z1>0 & Z1<=layers(1).t)=(roHfO2+1575).*exp(-(2.5-(2-
xdens1(Z1>0&Z1<=layers(1).t))).^6)+roHf;
 roy(Z2>0 & Z2<=layers(1).t)=(roHfO2+1575).*exp(-(2.5-(2-
xdens2(Z2>0&Z2<=layers(1).t))).^6)+roHf;
 roz(Z3>0 & Z3<=layers(1).t)=(roHfO2+1575).*exp(-(2.5-(2-
xdens3(Z3>0&Z3<=layers(1).t))).^6)+roHf;

%-------------------used for plotting 1-D plot of resisitvity------------------

 rox_=squeeze(rox((length(yspacing)-1)/2,(length(xspacing)-1)/2,:));
 xdens3_=squeeze(xdens3((length(yspacing)-1)/2,(length(xspacing)-1)/2,:));
 CumSUM=cumsum(xdens3(:,11:35,1));
 CumS=cumsum(rox(:,11:35,1));
 avg_denisty_in_centerCF=CumSUM(end)/(35-11);
 avg_resisitvity_in_centerCF=CumS(end)/(35-11);

 if nstep==1
 figure;
 semilogy(2-xdens3_,roz_)
title('Stochiometry X=2-i, where HfOX');
 hold off;
 % keyboard;
 end

 figure;
 plot(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(:,15),T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(:,12));
title('I vs V_25C');
 pause(.1);

------------------------
% pause(1);
% keyboard;
%-------------------------------------------------------------------------%
n=zeros(size(xmesh));
x=zeros(size(xmesh));

[n(zmesh>=0&zmesh<=layers(1).t)]=nmap(VacMap0,xmesh(zmesh>=0&zmesh<=layers(1).t)
,ymesh(zmesh>=0&zmesh<=layers(1).t),zmesh(zmesh>=0&zmesh<=layers(1).t),xspacing
,yspacing,zspacing);

Nvac=trapz(yspacing,trapz(xspacing,trapz(zspacing,n,3),2),1);

kthx(Z1>0&Z1<=layers(1).t)=(kthHf).*exp(-(2.5-
(xdens1(Z1>0&Z1<=layers(1).t))).^6)+kthHfO2;
ktthy(Z2>0&Z2<=layers(1).t)=(kthHf).*exp(-(2.5-
(xdens2(Z2>0&Z2<=layers(1).t))).^6)+kthHfO2;
ktthz(Z3>0&Z3<=layers(1).t)=(kthHf).*exp(-(2.5-
(xdens3(Z3>0&Z3<=layers(1).t))).^6)+kthHfO2;

% - Passing in Voltage on ReRAM, and respective kth, ro, alpha, Te
% - SOR function
if Vreram>=0
    [V T I P Jinf Jsup] = DC_Sim_Drift_T(xspacing,yspacing,zspacing,0,Vreram,Te,Te,rox,roy,roz,kthx,kthy,
    kthz,Rinf,Rsup,Rthinf,Rthsup,alpha_T,V,T);
end

if Vreram<0
    [V T I P Jinf Jsup] = DC_Sim_Drift_T(xspacing,yspacing,zspacing,-Vreram,0,Te,Te,rox,roy,roz,kthx,kthy,
    kthz,Rinf,Rsup,Rthinf,Rthsup,alpha_T,V,T);
end

% Using the returned potential for the SOR to cal. Efield
Edev.Xmesh=xmesh; Edev.Ymesh=ymesh; Edev.Zmesh=zmesh;
[Edev.Exmesh,Edev.Eymesh,Edev.Ezmesh]=gradient(V,xspacing(2)-xspacing(1),yspacing(2)-yspacing(1),zspacing(2)-zspacing(1));

Edev.Exmesh=-Edev.Exmesh; Edev.Eymesh=-Edev.Eymesh; Edev.Ezmesh=-Edev.Ezmesh;
Emag=sqrt((Edev.Exmesh).^2 + (Edev.Eymesh).^2 + (Edev.Ezmesh).^2);

% Using the returned potential for the SOR to cal. Local Temperature
Tdev.Xmesh=xmesh; Tdev.Ymesh=ymesh; Tdev.Zmesh=zmesh;
Tdev.Tmesh=T;
nrec_previous=nrec; % used to track recombination

%---------------------------------kMC-----------------------------------------%
% - DriftDiffusion of ions/vac is calculated using kMC method
% - returns time, mpas_vs_ t--> a struct of event positions (3D).
% - An event is described as either generation of a vacancy/ion pair,
%   recombinations, or diffusion.
% - Diffusion takes into account External Efeild + Colombic replusion,
%   and local Temperature, with Ea values of 0.7eV
% - Recombination has activation energy of .2eV
% - Generation has 3 respective activation energies pending on if
%   clusters, in GB, or Bulk
I=abs(I);

if I>200e-9
    restriction=0;
end

%---------------------------------kMC-----------------------------------------%
if FormingFlag==1 ;
    if (compl==1)
        VFlag=0;
    end
if (abs(InCF_Precentages(end-1)-InCF_Precentages(end))<.00005)
&& (InCF_Precentages(end)<100 && InCF_Precentages(end-1)<100) && (sexflag==1)
    R_now=R_now+1 %cuttoff
end

elseif ( (compl==0) && (abs(rCF_array(end-1)==rCF_array(end))) ) %
if rad of CF has not changed
    VFlag=0;
    if ( abs(((InCF_Precentages(end-1))-InCF_Precentages(end))<.00005) && (InCF_Precentages(end)<100 && InCF_Precentages(end-1)<100) && sexflag==1 )
        R_now=R_now+1 %cuttoff
    end
else
endif % if radius of CF has changed
    VFlag=0;
    if sexflag==1 && NoRecomb==1 && abs(Vtotal)==0 % if noRecomb is true
        then enter
            R_now=R_now+1;
        end

        if R_now>=3 && abs(Vtotal)>0
            R_now=1;
        end
    end
%---------------------------
if ResetFlag==1 || SetFlag==1
    VFlag=0;
    if sexflag==1 && NoRecomb==1 && abs(Vtotal)==0 % if noRecomb is true
        then enter
            R_now=R_now+1;
        end

        if R_now>=3 && abs(Vtotal)>0
            R_now=1;
        end
    end
%---------------------------
if Vreram==0
    VpinChargeFlag=2;
end
if Vreram>0 || Vreram<0
    VpinChargeFlag=1;
end
NoRecomb=0; % re-initialize
R_now
[ttime maps_vs_t_, transitions trec rec_pos nrec tSpacer Spacer_pos nSpacer ngen lifetime VacMap2_new In sexflag rCF]=Ions_DD_kMC(TurnOFFGEN_Recomb_STOP,VpinChargeFlag,SetFlag,FormingFlag,ResetFlag,sexflag,Rcutoff(R_now),VFlag,PULSE_OFF,IcomplMeet,restriction,1,
IonMap,VacMap0, VacMap2,Edev,Tdev,min(tsim_max-tottime,tin(nVin+1)-tottime),10000, trec, rec_pos, nrec, tSpacer, Spacer_pos, nSpacer);

if numel(maps_vs_t_(end).RecombPos)==numel(RecombMap) % this there was no change
    NoRecomb=1; % then NoRecomb is true
end
%---------------------------------------------------------------------------data collection from kMC---------------------------------------%
% keyboard;
InCF_Precentages=[InCF_Precentages In];
rCF_array=[rCF_array rCF];
VacMap2=VacMap2_new;
tottime=tottime+time(end);
VPAtottime=VPAtottime+vpa(time(end),40);

VacMap0=maps_vs_t_(end).Vac;
IonMap=maps_vs_t_(end).Ions;
RecombMap=maps_vs_t_(end).RecombPos;
SpacerIonMap=maps_vs_t_(end).SpacerIonsPos;

CountIonsInTE=0;
CountIonsInMeOx=0;
CountIonsInSpacer=0;
IonMapTemp=IonMap;
ngen_count=ngen_count+ngen;

parfor xx=1:length(IonMap(:,1))
   if IonMapTemp(xx,3)> 5e-9
      CountIonsInTE=CountIonsInTE+1;
   end
   if IonMapTemp(xx,3)< 5e-9
      CountIonsInMeOx=CountIonsInMeOx+1;
   end
end

%-------------------------------------------------------------------------%
%--------------simulating what happens when Icompl is reached--------------%
Rreram=abs(Vreram/I);
%-------------------------------------------------------------------------%
%---- Recording data for plotting and screen-output------------------------%
% Collection of transient information recorded in an ordered array
%Use the variable name to understand what variable is recorded where.
%totTime_density_resitivty_#vacInMeOx
/ionInMeOx #ionInTE_tot#Recomb #Generation #IonSpacer_Temp_Efeild_Current
V(onReRAM)_V(onLoadR)_totalV_VssReRAM_VssLoadR_Resistance(onReRAM)_Resistance(onLoadR)_ParCap_tau

T_x_ro_Nv_NMO_NTE_NR NG NS T_E I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(nstep,:)=
[TotTime_density_resitivty_#vacInMeOx
/ionInMeOx #ionInTE_tot#Recomb #Generation #IonSpacer_Temp_Efeild_Current
V(onReRAM)_V(onLoadR)_totalV_VssReRAM_VssLoadR_Resistance(onReRAM)_Resistance(onLoadR)_ParCap_tau]

if compl==1;
   Aft_time=TT_x_ro_Nv_NMO_NTE_NR NG NS T_E I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L(end,1)
   - TT_x_ro_Nv_NMO_NTE_NR NG NS T_E I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L(NStepFrame_where
   eAfterIcompl_timer_starts(I));end;
   fprintf(\nGen_Bulk Ea %d, beta %d\n',Eact_genBULK,b_genBULK); % resistaivty is assumed to be Hf like at Hf0%d
   fprintf(\nGen GB Ea %d, beta %d\n',Eact_genGB,b_genGB); % resistaivty is assumed to be Hf like at Hf0%d
   fprintf(\nRecomb Ea %d, Eform); % resistaivty is assumed to be Hf like at Hf0%d
   fprintf(' Avg x of CF center,20% of HfO2-x --> 2-x=1.55, 2-x=%d 
',2-avg_density_in_centerCF);
   fprintf(' Avg Rho of CF center, rho=%d \n
',avg_resisitvity_in_centerCF);
   fprintf(' MaxTemp=%d, \n
',max(T(:)));
   fprintf(' MaxEfield(z dir)=%d, \n
',min(Edev.Ezmesh(:)));
   fprintf(' I=%d, \n
',I);
   fprintf(' V(on ReRAM)=%d, \n
',Vreram);
   fprintf(' V(on LoadR)=%d, \n
',Vload);
   fprintf(' VssReRAM=%d, \n
',VssReRAM);
   fprintf(' VssLoadR=%d \n
',VssLoadR);

179
fprintf(' Vtotal=%d,

',Vtotal);
fprintf(' Tau(RC delay)=%d,

',tau);
fprintf(' Cp=%d,

',Cp);
fprintf(' R(on ReRAM)=%d,

',Rreram);
fprintf(' R(on LoadR)=%d,

',Rload);
fprintf(' Icompl=%d,

',Icompl);
fprintf(' NumVac in MeOx=%d,

',Nvac);
fprintf(' NumIons in MeOx=%d,

',CountIonsInMeOx);
fprintf(' NumIons in OEL=%d,

',CountIonsInTE);
fprintf(' NumIons in Spacer=%d, %dSpacer-1);%tot num of pairs deleted
leaving thethur side
fprintf(' NumRecomb Events=%d, %d,nrec-1); %tot num of pairs deleted
from recombination
fprintf(' NumGenera Events=%d, %d,ngen_count);%tot num of pairs deleted
leaving thethur side
toc;

nameXX=strcat(cd,'/Output/Pmap_n',num2str(nstep));
nameXX=strcat(cd,'/Output/Tmap_n',num2str(nstep));
nameXX=strcat(cd,'/Output/Vmap_n',num2str(nstep));
nameXX=strcat(cd,'/Output/VacMap0_n',num2str(nstep));
nameXX=strcat(cd,'/Output/VacMap2_n',num2str(nstep));
nameXX=strcat(cd,'/Output/IonMap_n',num2str(nstep));
nameXX=strcat(cd,'/Output/RecombMap_n',num2str(nstep));
nameXX=strcat(cd,'/Output/SpacerIonMap_n',num2str(nstep));

% Save power map (binary format to save HD space)
% Save temperature map
% Save voltage map
% Save vacancy map
% Save ion map
% Save Recomb map
% Save Spacer Map
% Save transient information
nameXX=strcat(cd,'/Output/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr
NC_Cp_u_L');

fid=fopen(nameXX,'wb');
fwrite(fid,T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc,'double');
fclose(fid);

VPAt(nstep)=VPAtottime;
t(nstep)=tottime;

if I >= 5e-6 && cheat_5e_neg6==0;
savedNstep5e6=nstep;
save('Severything5e-6');
cheat_5e_neg6=1;
end

if I >= 9e-6 && cheat_9e_neg6==0;
savedNstep9e6=nstep;
save('Severything9e-6');
cheat_9e_neg6=1;
end

if I >= 19e-6 && cheat_19e_neg6==0;
savedNstep19e6=nstep;
save('Severything19e-6');
cheat_19e_neg6=1;
end

if I >= 29e-6 && cheat_29e_neg6==0;
savedNstep29e6=nstep;
save('Severything29e-6');
cheat_29e_neg6=1;
end

if I >= 39e-6 && cheat_39e_neg6==0;
savedNstep39e6=nstep;
save('Severything39e-6');
cheat_39e_neg6=1;
end

if I >= 49e-6 && cheat_49e_neg6==0;
savedNstep49e_neg6=nstep;
save('Severything49e_neg6');
cheat_49e_neg6=1;
end

if I >= 75e-6 && cheat_75e_neg6==0;
savedNstep75e6=nstep;
save('Severything75e-6');
cheat_75e_neg6=1;
end

if I >= 99e-6 && cheat_99e_neg6==0;
savedNstep99e6=nstep;
save('Severything99e-6');
cheat_99e_neg6=1;
end

% if compl==1 && roundn(VssReRAM,-3)==roundn(Vreram,-3) && I>=Icompl
% FlagAll=0; % turns off recomb and generation
save('VssEqualsVreram_RGD_On');

% this is a hardrive space saver and time saver, disables recomb
% and gen after Vss and V settles and Icompl is meet
% however, I dont think it is correct to just turn off recomb and
gen

if I >= Icompl && compl==0 && (FormingFlag==1 || SetFlag==1);
%Compliance reached
 save('IcomplReached1');
 keyboard;
 compl=1;
end

if I <= 1e-9 && ResetFlag==1 && abs(Vtotal)==abs(ResetPH)
%Compliance reached
 save('ResetDone');
 keyboard;
 compl=1;
end

if PULSE_OFF  %if Vramp is on
 % keyboard;
 if Vtotal==0 && roundn(tottime,-12)>=roundn((tin(nVin+1)),-12)
  fprintf('Step increase__
');
  nVin=nVin+1;
  if FormingFlag==0 && onetime==1
   save('beforeReset_AfterRelaxation');
   % keyboard;
   onetime=0;
   fprintf('Temp set to 300, just in case temp was dif during
relaxation\n');
   T=300*ones(length(yspacing),length(xspacing),length(zspacing));
  end
  Vtotal=Vin(nVin);
  nrec_temp=0;
elseif abs(Vtotal)>0 && roundn(tottime,-12)>=roundn((tin(nVin)),-12)
  fprintf('Step increase__
');
  nVin=nVin+1;
  Vtotal=Vin(nVin);
  nrec_temp=0;
end
% if the almost never ending pattern of gen followed by recomb happens
% (for the sake of saving time -->nVin is skipped
if (nrec_temp > 40) && (CountIonsInMeOx < 10)
  nVin=nVin+1;
  Vtotal=Vin(nVin-1);
  fprintf('Step increase\n');
  fprintf('skipping step for time, 40 cycles for Gen and recomb\n');
  Vtotal=Vin(nVin);
  tottime=tin(nVin);
  VPAt(nstep)=VPAtottime;
  t(nstep)=tottime;
  nrec_temp=0;
end
end

nstep=nstep+1;

if nrec==nrec_previous+1  % tracks recombination
  nrec_temp=nrec_temp+1;

}
end

close all;
end
save('EveryEverythingyo4');

I=20.5e-6

%Start_num=VOLTOFF_nstep-1;
Start_num=1;
%Stop_num=length(TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_Rl_Cp_u_L)-1;
Stop_num=length(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc);
tanimation=100;
frameRate=length(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc)/(tanimation);

mkdir('ILoveLiz');

%%
1100:1:1110 1200:1:1210 1250:50:5000
5000:20:length(TT_x_ro_Nv_NMOx_NTE_NR_NSp_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_Rl_Cp_tu)];
%%
%%V=[1:1:length(TT_x_ro_Nv_NMO_x_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_Rl_Cp_tu)]
%%
%%V=[Start_num:10:NStepFrame_where_AfterIcompl_timer_starts-1
%%NStepFrame_where_AfterIcompl_timer_starts-1:1:Stop_num-1];
%%
%%V=[Start_num:1:Stop_num];

NStepFrame_where_AfterIcompl_timer_starts=1;

plotCountOutR(InCF_Precentages,xspacing,yspacing,zspacing,roz,frameRate,'ILoveLiziPooh',V, Rload,1,PULSE_OFF, Stop_num,Start_num, Icompl,NStepFrame_where_AfterIcompl_timer_starts);

Copy_of_ALL_IMW_DDanimation3(InCF_Precentages,xspacing,yspacing,zspacing,roz,frameRate,'ILoveLizi',V, Rload,1,PULSE_OFF, Stop_num,Start_num, Icompl,NStepFrame_where_AfterIcompl_timer_starts);

V=[1:10:length(TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_Rl_Cp_u_L)];

DDanimation3(xspacing,yspacing,zspacing,roz,frameRate,'Vramp_forming2.avi',V, Rload,vramp_rate,PULSE_OFF,t_after_Icompl);

1100:1:1110 1200:1:1210 1250:50:5000
5000:20:length(TT_x_ro_Nv_NMOx_NTE_NR_NSp_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_Rl_Cp_tu)];

DDanimation5(xspacing,yspacing,zspacing,roz,frameRate,'Vramp_forming_2_Vreram0 atIcompl.avi',V, Rload,vramp_rate,PULSE_OFF,t_after_Icompl, Icompl,Stop_num,Start_num);

figure;
fighist(transitions(:,1),100);
xlabel('# ion Ions','FontSize',16);
ylabel('# of steps (transactions)','FontSize',16);

figure;
fighist(transitions(:,2),100);
xlabel('Angle Phi','FontSize',16);
ylabel('# of steps (transactions)','FontSize',16);
% figure;
% hist(transitions(:,3),100);
% xlabel('Angle Theta','FontSize',16);
% ylabel('# of steps (transactions)','FontSize',16);

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

function InitializeDevice()

global flags;
global layers;

% fid=fopen('Simulation parameters\par_set.txt', 'r');
% par_set=textscan(fid, '%f');
% folder='Simulation parameters\SimParInterface';
flags = struct('WKB',0,'POISSON',0,'BIDIRECTIONS',0,'HOLES',0,'MULTITRAP_TRANSITIONS',0,
                 'INF_EXCHANGE',0,'SUP_EXCHANGE',0,'N_2D',0,'EMISSION_GROUND_STATE',0,'MULTIPHONON_QUANT',0,
                 'MULTIPHONON_FIELD_INDUCED_TRANSITION',0,'MULTIPHONON_NON_ADIABATIC_TRANSITION',0,
                 'POISSON_DT',0,'S_DISPERSION',0);

inf=struct('name','','me',0,'mh',0,'WF',0,'Nv',0,'Kth',0,'phib',0);
Si_par=struct('k',0,'me',0,'mh',0,'me0',0,'me1',0,'md0',0,'md1',0,'mh0',0,'mh1',0,
               'mdh0',0,'mdh1',0,'g0',0,'g1',0,'gh0',0,'gh1',0,'Nc0',0,'Nv0',0,'Kth',0,'N00',
               'N10',0,'N01',0,'Nh00',0,'Nh10',0,'Nh11',0,'ki',0);
sup=struct('name','','me',0,'mh',0,'WF',0,'Nv',0,'Kth',0,'phib',0);
keyboard;
% load stack parameters
fid=fopen('device.txt','r');
scanf(fid, '%s',1);
device_type = fscanf(fid, '%s',1);
scanf(fid, '
',1);
scanf(fid, '%s',1);

184
inf.name = fscanf(fid, '%s',1);
fscanf(fid, '%s',1);
layernames{1} = fscanf(fid, '%s',1);
layernames{2} = fscanf(fid, '%s',1);
layert(1) = fscanf(fid, '%f',1);
layert(2) = fscanf(fid, '%f',1);
Nsub = fscanf(fid, '%f',1);
DeltaVfb_Qit = fscanf(fid, '%f',1);
fscanf(fid, '
',1);
fscanf(fid, '%s',1);
fscanf(fid, '
',1);
fclose(fid);
if strcmp(device_type,'MIM')
  METAL_INF=1;
  if DeltaVfb_Qit~=0
    DeltaVfb_Qit=0;
    warning('DeltaVfb_Qit ignored, since Qit not effective to DVFB in a MIM device: DVFB set to 0');
  end
else
  METAL_INF=0;
end
METAL_SUP=1;
k=zeros(1,length(layer_names));
phioff=zeros(1,length(layer_names));
me=zeros(1,length(layer_names));
mh=zeros(1,length(layer_names));
Eg=zeros(1,length(layer_names));
hw0=zeros(1,length(layer_names));
Kth=zeros(1,length(layer_names));
Kv=zeros(1,length(layer_names));
Eact=zeros(1,length(layer_names));
p0=zeros(1,length(layer_names));
Lbond=zeros(1,length(layer_names));
mol_dens=zeros(1,length(layer_names));
mred=zeros(1,length(layer_names));
Nv=zeros(1,length(layer_names));
phioff_VB=zeros(1,length(layer_names));
for i=1:length(layer_names)
  if layert(i)>0
    name1xx=strcat(layer_names{i},'.txt');
    [names v]=ReadParameterFile(name1xx);
    k(i)=v(1);
    phioff(i)=v(2);
  end
end
me(i)=v(3)*m0;
mh(i)=v(4)*m0;
Eg(i)=v(5);
hw0(i)=v(6);
Kth(i)=v(7);
Kv(i)=v(8);
Eact(i)=v(9);
p0(i)=v(10);
Lbond(i)=v(11);
mol_dens(i)=v(12);
mred(i)=v(13);
Nv(i)=v(14);
phioff_VB(i)=Eg(i)-phioff(i)-Eg_Si(300);
end
end
layers =
struct('name',layernames,'t',num2cell(layert),'k',num2cell(k),'phioff',num2cell
(phioff), 'phioff_VB', num2cell(phioff_VB), 'me', num2cell(me), 'mh'...
, num2cell(mh), 'Eg', num2cell(Eg), 'hw0', num2cell(hw0), 'Kth', num2cell(Kth), 'Kv', nu
m2cell(Kv), 'Eact', num2cell(Eact), 'p0', num2cell(p0)...,
'Lblond', num2cell(Lbond), 'mol_dens', num2cell(mol_dens), 'mred', num2cell(mred), 'N
v', num2cell(Nv));
% load electrode parameters
if(METAL_INF == 1)
    namelxx=strcat(inf.name,'.txt');
    [names v] =  ReadParameterFile(namelxx);
    inf.me=v(1)*m0;
    inf.WF=v(2);
    inf.Nv=v(3);
    inf.Kth=v(4);
    inf.phib=inf.WF-Si_par.ki+layers(1).phioff;
    inf.phibVB=layers(1).Eg-sup.phib;
else
    inf.me=Si_par.me;
    inf.mh=Si_par.mh;
    inf.Kth=Si_par.Kth;
    inf.phib=layers(1).phioff;
    inf.phibVB=layers(1).Eg-sup.phib;
end
if(METAL_SUP == 1)
    namelxx=strcat(sup.name,'.txt');
    [names v] =  ReadParameterFile(namelxx);
    sup.me=v(1)*m0;
    sup.WF=v(2);
    sup.Nv=v(3);
    sup.Kth=v(4);
    sup.phib=sup.WF-Si_par.ki+conditional(layers(1).t>0,layers(1).phioff,layers(2).phioff);
    sup.phibVB=conditional(layers(1).t>0,layers(1).Eg,layers(1).Eg)-sup.phib;
else
    sup.me=Si_par.me;
    sup.mh=Si_par.mh;
    sup.Kth=Si_par.Kth;
    sup.phib=conditional(layers(2).t>0,layers(2).phioff,layers(1).phioff);
    sup.phibVB=conditional(layers(2).t>0,layers(2).Eg,layers(1).Eg)-sup.phib;
end
dev_string=[device_type,'_',inf.name,'_',num2str(layers(1).t*1e9),'_nm',layers(1).name,'_'];
if layers(2).t>0
    dev_string=[dev_string,num2str(layers(2).t*1e9),'_nm',layers(2).name,'_'];
end
dev_string=[dev_string,sup.name];
% load flags
[names v]=ReadParameterFile('flags.txt');
flags.WKB = v(1);
flags.POISSON= v(2);
%flags.SELF_CONSISTENT= v(3);
flags.BIDIRECTIONS= v(3);
flags.HOLES= v(4);
flags.MULTITRAP_TRANSITIONS= v(5);
flags.INF.Exchange= v(6);
flags.SUP.Exchange= v(7);
flags.N_2D= v(8);
flags.EMISSION.GROUND.STATE= v(9);
flags.MULTIPHONON_QUANT = v(10);
flags.MULTIPHONONFIELD.INDUCED_TRANSITION=v(11);
flags.MULTIPHONON_NON_ADIABATIC_TRANSITION=v(12);
flags.POISSON_DT=v(13);
flags.S_DISPERSION=v(14);
flags.COHERENT=v(15);
flags.CHARGE_SHEET_POISSON=v(16);
flags.CAPTURE=v(17);
flags.EMISSION=v(18);
flags.EREL.RED=v(19);
flags.COULOMB_ANALYTIC=v(20);

% load geometry parameters
[names v] = ReadParameterFile('geometry.txt');
X1_pot = v(1);  X2_pot = v(2);  Y1_pot = v(3);  Y2_pot = v(4);
X1_T = v(5);  X2_T = v(6);  Y1_T = v(7);  Y2_T = v(8);
X1_D = v(9);  X2_D = v(10);  Y1_D = v(11);  Y2_D = v(12);

% load Numerical simulation parameters
[names v]= ReadParameterFile('Numericalsims_parameters.txt');
numsims_par.zstep_tunn=v(1);

if v(2)>(X2_T-X1_T)/2
    warning('Bad temperature mesh dimensions: X cell length fixed at %d',(X2_T-
    X1_T)/2);
end
if v(3)>(Y2_T-Y1_T)/2
    warning('Bad temperature mesh dimensions: Y cell length fixed at %d',(Y2_T-
    Y1_T)/2);
end
if v(4)>(sum([layers(:,t)])/2
    warning('Bad temperature mesh dimensions: Z cell length fixed at %d',
    (sum([layers(:,t)]))/2);
end
numsims_par.lregX_T=conditional(v(2)<(X2_T-X1_T)/2,v(2),(X2_T-X1_T)/2);
numsims_par.lregY_T=conditional(v(3)<(Y2_T-Y1_T)/2,v(3),(Y2_T-Y1_T)/2);
numsims_par.lregZ_T=conditional(v(4)<(sum([layers(:,t)]))/2,v(4),(sum([layers(:,
    t)]))/2);
numsims_par.n_max_cycles_T=v(5);
numsims_par.varMax_T=v(6);
if flags.CHARGE_SHEET_POISSON
    % Potential discretization only along the z direction
    v(7)=(X2_pot-X1_pot)/2;
v(8)=(Y2_pot-Y1_pot)/2;
end
if v(7)>(X2_pot-X1_pot)/2
    warninig('Bad potential mesh dimensions: X cell length fixed at %d',(X2_pot-
    X1_pot)/2);
end
if v(8)>(Y2_pot-Y1_pot)/2
    warninig('Bad potential mesh dimensions: Y cell length fixed at %d',(X2_pot-
    X1_pot)/2);
if v(9)>(sum([layers(:,t)])/2)
    warning('Bad potential mesh dimensions: Z cell length fixed at %d',(sum([layers(:,t)])/2));
end
numsims_par.lregX_pot=conditional(v(7)<(X2_pot-X1_pot)/2,v(7),(X2_pot-X1_pot)/2);
numsims_par.lregY_pot=conditional(v(8)<(Y2_pot-Y1_pot)/2,v(8),(Y2_pot-Y1_pot)/2);
numsims_par.lregZ_pot=conditional(v(9)<(sum([layers(:,t)])/2),v(9),(sum([layers(:,t)])/2)/2);
numsims_par.n_max_cycles_V=v(10);
numsims_par.varMax_SOR=v(11);
numsims_par.n_max_cycles_I=v(12);
numsims_par.varMax_I=v(13);
numsims_par.zstep_map=v(14);
numsims_par.Estep=v(15);
numsims_par.Sstep=v(16);
numsims_par.lregX_D=conditional(v(17)<(X2_D-X1_D)/2,v(17),(X2_D-X1_D)/2);
numsims_par.lregY_D=conditional(v(18)<(Y2_D-Y1_D)/2,v(18),(Y2_D-Y1_D)/2);
numsims_par.lregZ_D=conditional(v(19)<(sum([layers(:,t)])/2),v(19),(sum([layers(:,t)])/2)/2);

% Device discretization (potential calculation (3D Poisson equation))
Xp=linspace(X1_pot,X2_pot,floor((X2_pot-X1_pot)/numsims_par.lregX_pot)+1);
YP=linspace(Y1_pot,Y2_pot,floor((Y2_pot-Y1_pot)/numsims_par.lregY_pot)+1);
ZP=linspace(0,sum([layers(:,t)]),floor((sum([layers(:,t)])/numsims_par.lregZ_pot)+1));

% Device discretization (temperature calculation (3D Fourier equation))
XT=linspace(X1_T,X2_T,floor((X2_T-X1_T)/numsims_par.lregX_T)+1);
YT=linspace(Y1_T,Y2_T,floor((Y2_T-Y1_T)/numsims_par.lregY_T)+1);
ZT=linspace(0,sum([layers(:,t)]),floor((sum([layers(:,t)])/numsims_par.lregZ_T)+1));

% Device discretization (defect generation, diffusion)
Xd=linspace(X1_D,X2_D,floor((X2_D-X1_D)/numsims_par.lregX_D));
Yd=linspace(Y1_D,Y2_D,floor((Y2_D-Y1_D)/numsims_par.lregY_D));
Zd=linspace(0,sum([layers(:,t)]),floor((sum([layers(:,t)])/numsims_par.lregZ_D)+1));

% Device discretization EZ map
% z discretization
Zmap=linspace(0,sum([layers(:,t)]),floor((sum([layers(:,t)])/numsims_par.zstep_map)+1));

% E discretization
max_Ec=max(inf.phib,sup.phib)+4;
min_Ev=min(-inf.phibVB,sup.phibVB)-4;
Emap=linspace(min_Ev,max_Ec,floor((max_Ec-min_Ev)/numsims_par.Estep+1));

% load defect distributions parameters
[distr,zmin,zmax]=ReadDefDistrFile('defect_distributions.txt');
for i=1:length(distr)
    name= strcat(distr(i),'.txt');
    [names v] = ReadParameterFile(name);
    if zmax(i)>sum([layers(:,t)]) || zmin(i)<0
        error(['Error: defect distributions are outside device boundaries: change file defect_distributions.txt']);
    end
    if zmax(i)>layers(1).t && zmin(i)<layers(1).t
        error(['Error: defect distribution ',distr(i),' across two different layers']);
    end
sigma_t_{i}=v(1);
charge_{i}=v(2);
Et_{min}_{i}=v(3);
Et_{max}_{i}=v(4);
deltaS=v(8);
if flags.S\_DISPERSION==1&&deltaS>0
  S_{min}_{i}=conditional(v(5)-\delta S/2>1,v(5)-\delta S/2,1);
  S_{max}_{i}=v(5)+\delta S/2;
else
  S_{min}_{i}=v(5); S_{max}_{i}=v(5);
end
ro_{i}=v(6);
def_type_{i} = v(7);
nt_{i}=v(6)*1e6*(X2\_D-X1\_D)*(Y2\_D-Y1\_D)*(z_{max}_{i}-z_{min}_{i});
gen_bondbreak_{i}=v(9);
Eg=conditional(z_{min}_{i}<=layers(1).t,layers(1).Eg,layers(2).Eg);
hw0=conditional(z_{min}_{i}<=layers(1).t,layers(1).hw0,layers(2).hw0);
meff=conditional(z_{min}_{i}<=layers(1).t,conditional(def_type_{i}==1,layers(1).me,layers(1).mh),conditional(def_type_{i}==1,layers(2).me,layers(2).mh));
sigma_t\_SRH_{i}=sigma_t\_SRH(sigma_t_{i},Eg,meff,v(5),hw0);
rt_{i}=10^{-2}*sqrt(sigma_t_{i}/\pi);
end
if isempty(distr)
distr=cell(1,0);
sigma_t=cell(1,0);
sigma_t\_SRH_=cell(1,0);
rt=cell(1,0);
charge=cell(1,0);
Et_{min}=cell(1,0);
Et_{max}=cell(1,0);
S_{min}=cell(1,0);
S_{max}=cell(1,0);
ro=cell(1,0);
def_type=cell(1,0);
z_{min}=cell(1,0);
z_{max}=cell(1,0);
nt=cell(1,0);
gen_bondbreak=cell(1,0);
end
def_distr =
struct('name',distr,'sigma_t',sigma_t,'sigma_t\_SRH',sigma_t\_SRH_,'rt',rt,'charge',charge,'Et_{min}',Et_{min},'Et_{max}',Et_{max},'S_{min}',S_{min},'S_{max}',S_{max},'ro',ro,'def_type',def_type,'z_{min}',z_{min},'z_{max}',z_{max},'nt',nt,'gen_bondbreak',gen_bondbreak);

EZmapsims_par=struct;
% load EZ map sims parameters
[names v] =  ReadParameterFile('EZmapsims_parameters.txt');
EZmapsims_par.RATES\_CALC\_NDEF=v(1);

Transientsims_par=struct;
% load Transient sims parameters
[names v] =  ReadParameterFile('Transientsims_parameters.txt');
Transientsims_par.var\_recalc=v(1);
Transientsims_par.POISSON\_RECALC=v(2);
Transientsims_par.T\_VAR=v(3);
Transientsims_par.T\_VAR\_EXACT=v(4);
Transientsims_par.DEF\_GEN=v(5);
Transientsims_par.T\_TRANSIENT=v(6);
Transientsims_par.Ft_SS=v(7);
Transientsims_par.nsteps_max=v(8);
Transientsims_par.ndef_max=v(9);
Transientsims_par.ION_MOTION=v(10);

% load Transient sims parameters
[names v] = ReadParameterFile('DCsims_parameters.txt');
DCsims_par.POT_VAR=v(1);

End
function [ PinnedVacMap , rad_pos] = grainboundries( theta_grain,
theta_variation, zspacing, precent_to_delete, precent_to_delete2, r_pinned, flag,
rad_pos, jj, jjj)
if flag==1
    PinnedVacMap=[];
    rad_pos=[];
    random_startpt=2*pi*rand(1,1);
    arc_length=theta_grain*r_pinned;
    min_rad_spacing=arc_length/r_pinned;
    %num_vac_pinned=floor(((2*pi)/min_rad_spacing))-1;
    for i=11:36
        rad_pos=linspace(0,2*pi,(2*pi)/min_rad_spacing);
        rad_pos=rad_pos(1:end-1);
        vary_sidewalls_possible=[rad_pos(1,1)+theta_variation rad_pos(1,1)
            rad_pos(1,1)-theta_variation]; %rad_pos is coming from previous for loop
        rad_pos=rad_pos+random_startpt;
        z_pinned=zspacing(i);
        Sub_PinnedVacMap=[(r_pinned*ones(size(rad_pos))).*cos(rad_pos);
            (r_pinned*ones(size(rad_pos))).*sin(rad_pos); z_pinned.*ones(size(rad_pos))];
    PinnedVacMap=vertcat(PinnedVacMap, Sub_PinnedVacMap);
end

XX=ceil(length(PinnedVacMap)*precent_to_delete);
for iii=1:XX
    A=randi(length(PinnedVacMap),1);
    PinnedVacMap=[PinnedVacMap(1:A-1,:);PinnedVacMap(A+1:end,:)];
end

if flag==0
    PinnedVacMap=[];
    arc_length=theta_grain*r_pinned;
    min_rad_spacing=arc_length/r_pinned;
    vary_sidewalls_possible=[rad_pos(1,1)+theta_variation rad_pos(1,1)]
    rad_pos(1,1)-theta_variation]; %rad_pos is coming from previous for loop
    X=randi(3,1);
    if X==1; random_startpt=vary_sidewalls_possible(1,1); end;
    if X==2; random_startpt=vary_sidewalls_possible(1,2); end;
    if X==3; random_startpt=vary_sidewalls_possible(1,3); end;
    num_vac_pinned=floor(((2*pi)/min_rad_spacing))-1;
    for l=1:36
        rad_pos=linspace(0,2*pi,(2*pi)/min_rad_spacing);
        rad_pos=rad_pos(1:end-1);
        if X==1; random_startpt=vary_sidewalls_possible(1,1); end;
        if X==2; random_startpt=vary_sidewalls_possible(1,2); end;
        if X==3; random_startpt=vary_sidewalls_possible(1,3); end;
        rad_pos=rad_pos+random_startpt;
        z_pinned=zspacing(i);
        Sub_PinnedVacMap=[(r_pinned*ones(size(rad_pos))).*cos(rad_pos);
        (r_pinned*ones(size(rad_pos))).*sin(rad_pos); z_pinned.*ones(size(rad_pos))]
        PinnedVacMap=vertcat(PinnedVacMap,Sub_PinnedVacMap);
    end

end

XX=ceil(length(PinnedVacMap)*precent_to_delete);
for iii=1:XX
    A=randi(length(PinnedVacMap),1);
    PinnedVacMap=[PinnedVacMap(1:A-1,:);PinnedVacMap(A+1:end,:)];
end

if flag==2
    PinnedVacMap=[];
    arc_length=theta_grain*r_pinned;
    min_rad_spacing=arc_length/r_pinned;
    vary_sidewalls_possible=[rad_pos(1,1)+theta_variation rad_pos(1,1)]
    rad_pos(1,1)-theta_variation]; %rad_pos is coming from previous for loop
    X=randi(3,1);
    if X==1; random_startpt=vary_sidewalls_possible(1,1); end;
    if X==2; random_startpt=vary_sidewalls_possible(1,2); end;
    if X==3; random_startpt=vary_sidewalls_possible(1,3); end;
    num_vac_pinned=floor(((2*pi)/min_rad_spacing))-1;
    for l=1:36
        rad_pos=linspace(0,2*pi,(2*pi)/min_rad_spacing);
        rad_pos=rad_pos(1:end-1);
        if X==1; random_startpt=vary_sidewalls_possible(1,1); end;
        if X==2; random_startpt=vary_sidewalls_possible(1,2); end;
        if X==3; random_startpt=vary_sidewalls_possible(1,3); end;
        rad_pos=rad_pos+random_startpt;
V=(rad_pos>=.5 & rad_pos<=1.3) | (rad_pos>=2 & rad_pos<=2.7)
| (rad_pos>=3.6 & rad_pos<=4.3) | (rad_pos>=5.1 & rad_pos<=6); %|
(rad_pos>=6.28 & rad_pos<=6.35)

z_pinned=zspacing(i);
Sub1=[(r_pinned+jj).*cos(rad_pos(V)); (r_pinned+jj).*sin(rad_pos(V));
z_pinned.*ones(size(rad_pos(V))])';
Sub2=[(r_pinned+jjj).*cos(rad_pos(~V));
(r_pinned+jjj).*sin(rad_pos(~V)); z_pinned.*ones(size(rad_pos(~V)))]';
PinnedVacMap=vertcat(PinnedVacMap,Sub1,Sub2);

XX=ceil(length(PinnedVacMap)*precent_to_delete);
for iii=1:XX
    A=randi(length(PinnedVacMap),1);
    PinnedVacMap=[PinnedVacMap(1:A-1,:);PinnedVacMap(A+1:end,:)];
end
end

%keyboard;
PinnedVacMap(PinnedVacMap>10e-9)=9.95e-9;
end

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
function [FVin Ftin RVin S1Vin S1tin R1Vin R1tin TotalVin Totaltin 
TotTimeForm TotTimeRelaxReset TotTimeRelaxSet1 TotTimeRelaxReset2 
TotTimeRelaxSet2 TotTimeRelaxReset3 TotTimeRelaxSet3 TotTimeRelaxReset4 
TotTimeRelaxSet4 TotTimeRelaxReset5 TotTimeRelaxSet5 TotTimeRelaxReset6 
TotTimeRelaxSet6 TotTimeRelaxReset7 TotTimeRelaxSet7 TotTimeRelaxReset8 
TotTimeRelaxSet8 TotTimeRelaxReset9 TotTimeRelaxSet9 TotTimeRelaxReset10 
TotTimeRelaxSet10 TotTimeRelaxReset11 TotTimeRelaxSet11 TotTimeRelaxReset12] = 
PulseTrain(Tr_Tf_TimeTotal,FormPW,FormPH,ResetPW,ResetPH,SetPW,SetPH,Relaxation 
Time) 
% note in real life pulse schemes, the (PW and rise/fall) time 
% #_of_dataPoints must be equal - here to save time in simulations the 
% #_of_dataPoints for (this) is not equal - this will cause each step in the 
% output I-V, if recomb/and gen are both on, to seem spikey --- to reduce this 
% need to avg 
%if Ic is reached (during Set and Forming) 
Ftr_NumDataPts=(Tr_Tf_TimeTotal*1e9)*5; %Forming Rise Time details 
Ftr_StartVin=0; 
Ftr_StopVin=FormPH; 
Ftr_Starttin=0; 
Ftr_Stoptin=Tr_Tf_TimeTotal; 
Ftr_Vin=linspace(Ftr_StartVin,Ftr_StopVin,Ftr_NumDataPts+1); 
Ftr_tin=linspace(Ftr_Starttin,Ftr_Stoptin,Ftr_NumDataPts+1); 

FPW_NumDataPts=(FormPW*1e9)*5; %Forming PW Time details 
FPW_StartVin=FormPH; 
FPW_StopVin=FormPH; 
FPW_Starttin=Tr_Tf_TimeTotal+(Ftr_tin(2)-Ftr_tin(1)); 
FPW_Stopitin=FPW_Starttin+FormPW; 
FPW_Vin=linspace(FPW_StartVin,FPW_StopVin,FPW_NumDataPts+1); 
FPW_tin=linspace(FPW_Starttin,FPW_Stopitin,FPW_NumDataPts+1); 
if roundn((FPW_tin(4)-FPW_tin(3)),12) ~= roundn((Ftr_tin(4)-Ftr_tin(3)),12) 
fprintf('Forming step for tr and PW is NOT the same'); 
end 
Ftf_NumDataPts=Ftr_NumDataPts*10; %Forming Fall Time details 
Ftf_StartVin=Ftr_StopVin; 

fprintf('Forming step for tr and PW is NOT the same');
Ftf_StopVin=Ftr_StartVin;
Ftf_Starttin=FPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Ftf_Stoptin=Ftf_Starttin;
Ftf_Vin=linspace(Ftf_StartVin,Ftf_StopVin,Ftf_NumDataPts+1);
Ftf_tin=linspace(Ftf_Starttin,Ftf_Stoptin,Ftf_NumDataPts+1);

Vin=[Ftr_Vin FPW_Vin Ftf_Vin];
tin=[Ftr_tin FPW_tin Ftf_tin];

FVin=Vin;
Ftin=tin;
% keyboard;
% figure;
% plot(Ftin,FVin);

TotTimeForm=tin(end);
%--------------------------------------------------------------------------
%Relaxation Time
Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];
%--------------------------------------------------------------------------
%1st Reset after Forming

Rtr_NumDataPts=Ftr_NumDataPts;  %1stReset Rise Time details
Rtr_StartVin=0;
Rtr_StopVin=ResetPH;
Rtr_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Rtr_Stoptin=Rtr_Starttin+(Tr_Tf_TimeTotal-0);%%%%%%%%%%%%%%%%%%%%%%%%%changed
Rtr_Vin=linspace(Rtr_StartVin,Rtr_StopVin,Rtr_NumDataPts+1);
Rtr_tin=linspace(Rtr_Starttin,Rtr_Stoptin,Rtr_NumDataPts+1);

RPW_NumDataPts=50;  %1st Reset PW Time details %%%%%%%%%%%%%changed from 1000
RPW_StartVin=ResetPH;
RPW_StopVin=ResetPH;
RPW_Starttin=Rtr_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
RPW_Stoptin=RPW_Starttin+ResetPW;
RPW_Vin=linspace(RPW_StartVin,RPW_StopVin,RPW_NumDataPts+1);
RPW_tin=linspace(RPW_Starttin,RPW_Stoptin,RPW_NumDataPts+1);

if roundn((RPW_tin(4)-RPW_tin(3)),-12) ~= roundn((Rtr_tin(4)-Rtr_tin(3)),-12)
fprintf('1st Reset after Forming ,step for tr and PW is NOT the same');
end

Rtf_NumDataPts=Rtr_NumDataPts; %1st Reset Fall Time details
Rtf_StartVin=Rtr_StopVin;
Rtf_StopVin=Rtr_StartVin;
Rtf_Starttin=RPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Rtf_Stoptin=Rtf_Starttin+(Tr_Tf_TimeTotal-0);%%%%%%%%%%%%%%%%%%%%%%%%%changed
Rtf_Vin=linspace(Rtf_StartVin,Rtf_StopVin,Rtf_NumDataPts+1);
Rtf_tin=linspace(Rtf_Starttin,Rtf_Stoptin,Rtf_NumDataPts+1);

RVin=[Rtr_Vin RPW_Vin Rtf_Vin];
Rtin=[Rtr_tin RPW_tin Rtf_tin];

Vin=[Vin RVin];
tin=[tin Rtin];
% figure;
% plot(Rtin,RVin);
TotTimeRelaxReset=tin(end);
%--------------------------------------------------------------------------
Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];

% Set1  
Str_NumDataPts=Ftr_NumDataPts;  % Forming Rise Time details  
Str_StartVin=0;  
Str_StopVin=SetPH;  
Str_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));  
Str_StopOpt=Str_Starttin+Tr_Tf_TimeTotal;  
Str_Vin=linspace(Str_StartVin,Str_StopVin,Str_NumDataPts+1);  
Str_tin=linspace(Str_Starttin,Str_StopOpt,Str_NumDataPts+1);

SPW_NumDataPts=1000;  % Forming Rise Time details  
SPW_StartVin=SetPH;  
SPW_StopVin=SetPH;  
SPW_StartOpt=Str_StopOpt+(Ftr_tin(2)-Ftr_tin(1));  
SPW_StopOpt=SPW_StartOpt+SetPW;  
SPW_Vin=linspace(SPW_StartVin,SPW_StopVin,SPW_NumDataPts+1);  
SPW_tin=linspace(SPW_StartOpt,SPW_StopOpt,SPW_NumDataPts+1);

if roundn((SPW_tin(4)-SPW_tin(3)),-12) ~= roundn((Str_tin(4)-Str_tin(3)),-12)
fprintf('1st Reset after Forming, step for tr and PW is NOT the same');
end

Stf_NumDataPts=Str_NumDataPts; % Forming Fall Time details  
Stf_StartVin=Str_StopVin;  
Stf_StopVin=Str_StartVin;  
Stf_StartOpt=SPW_StopOpt+(Ftr_tin(2)-Ftr_tin(1));  
Stf_StopOpt=Stf_StartOpt+Tr_Tf_TimeTotal;  
Stf_Vin=linspace(Stf_StartVin,Stf_StopVin,Stf_NumDataPts+1);  
Stf_tin=linspace(Stf_StartOpt,Stf_StopOpt,Stf_NumDataPts+1);

S1Vin=[Str_Vin SPW_Vin Stf_Vin];  
S1tin=[Str_tin SPW_tin Stf_tin];

Vin=[Vin S1Vin];  
tin=[tin S1tin];

% figure;  
% plot(S1tin,S1Vin);  
TotTimeRelaxSet1=tin(end);  

Vin=[Vin 0];  
tin=[tin tin(end)+RelaxationTime];

% 2nd Reset after Forming  

Rtr_NumDataPts=Ftr_NumDataPts;  % Forming Rise Time details  
Rtr_StartVin=0;  
Rtr_StopVin=ResetPH;  
Rtr_StartOpt=tin(end)+(Ftr_tin(2)-Ftr_tin(1));  
Rtr_StopOpt=Rtr_StartOpt+Tr_Tf_TimeTotal;  
Rtr_Vin=linspace(Rtr_StartVin,Rtr_StopVin,Rtr_NumDataPts+1);  
Rtr_tin=linspace(Rtr_StartOpt,Rtr_StopOpt,Rtr_NumDataPts+1);

RPW_NumDataPts=1000;  % Forming Rise Time details  
RPW_StartVin=ResetPH;  
RPW_StopVin=ResetPH;  
RPW_StartOpt=RPW_StopOpt+(Ftr_tin(2)-Ftr_tin(1));  
RPW_StopOpt=RPW_StartOpt+ResetPW;  
RPW_Vin=linspace(RPW_StartVin,RPW_StopVin,RPW_NumDataPts+1);  
RPW_tin=linspace(RPW_StartOpt,RPW_StopOpt,RPW_NumDataPts+1);
if roundn((RPW_tin(4)-RPW_tin(3)),-12) ~= roundn((Rtr_tin(4)-Rtr_tin(3)),-12)
fprintf('2st Reset after Forming, step for tr and PW is NOT the same');
end

Rtf_NumDataPts=Rtr_NumDataPts; % Forming Fall Time details
Rtf_StartVin=Rtr_StopVin;
Rtf_StopVin=Rtr_StartVin;
Rtf_Starttin=RPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Rtf_Stoptin=Rtf_Starttin+Tr_Tf_TimeTotal;
Rtf_Vin=linspace(Rtf_StartVin,Rtf_StopVin,Rtf_NumDataPts+1);
Rtf_tin=linspace(Rtf_Starttin,Rtf_Stoptin,Rtf_NumDataPts+1);

R1Vin=[Rtr_Vin RPW_Vin Rtf_Vin];
R1tin=[Rtr_tin RPW_tin Rtf_tin];

Vin=[Vin R1Vin];
tin=[tin R1tin];

figure;
plot(R1tin,R1Vin);

TotTimeRelaxReset2=tin(end);
figure;
plot(TotTimeRelaxReset2,TotalVin);
xlim([-5e-7 tin(end)+5e-7]);

Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];

Str_NumDataPts=Ftr_NumDataPts;  % Forming Rise Time details
Str_StartVin=0;
Str_StopVin=SetPH;
Str_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Str_Stoptin=Str_Starttin+Tr_Tf_TimeTotal;
Str_Vin=linspace(Str_StartVin,Str_StopVin,Str_NumDataPts+1);
Str_tin=linspace(Str_Starttin,Str_Stoptin,Str_NumDataPts+1);

SPW_NumDataPts=1000;  % Forming Rise Time details
SPW_StartVin=SetPH;
SPW_StopVin=SetPH;
SPW_Starttin=Str_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
SPW_Stoptin=SPW_Starttin+SetPW;
SPW_Vin=linspace(SPW_StartVin,SPW_StopVin,SPW_NumDataPts+1);
SPW_tin=linspace(SPW_Starttin,SPW_Stoptin,SPW_NumDataPts+1);

if roundn((SPW_tin(4)-SPW_tin(3)),-12) ~= roundn((Str_tin(4)-Str_tin(3)),-12)
fprintf('1st Reset after Forming, step for tr and PW is NOT the same');
end

Stf_NumDataPts=Str_NumDataPts; % Forming Fall Time details
Stf_StartVin=Str_StopVin;
Stf_StopVin=Str_StartVin;
Stf_Starttin=SPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Stf_Stoptin=Stf_Starttin+Tr_Tf_TimeTotal;
Stf_Vin=linspace(Stf_StartVin,Stf_StopVin,Stf_NumDataPts+1);
Stf_tin=linspace(Stf_Starttin,Stf_Stoptin,Stf_NumDataPts+1);

S1Vin=[Str_Vin SPW_Vin Stf_Vin];
S1tin=[Stf_tin SPW_tin Stf_tin];
Vin=[Vin S1Vin];
tin=[tin Sltin];

figure;
plot(S1tin,S1Vin);
TotTimeRelaxSet2=tin(end);

Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];

3rd Reset

Rtr_NumDataPts=Ftr_NumDataPts; %Forming Rise Time details
Rtr_StartVin=0;
Rtr_StopVin=ResetPH;
Rtr_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Rtr_Stopitin=Rtr_Starttin+Tr_Tf_TimeTotal;
Rtr_Vin=linspace(Rtr_StartVin,Rtr_StopVin,Rtr_NumDataPts+1);
Rtr_tin=linspace(Rtr_Starttin,Rtr_Stopitin,Rtr_NumDataPts+1);

RPW_NumDataPts=1000; %Forming Rise Time details
RPW_StartVin=ResetPH;
RPW_StopVin=ResetPH;
RPW_Starttin=Rtr_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
RPW_Stopitin=RPW_Starttin+ResetPW;
RPW_Vin=linspace(RPW_StartVin,RPW_StopVin,RPW_NumDataPts+1);
RPW_tin=linspace(RPW_Starttin,RPW_Stopitin,RPW_NumDataPts+1);

if roundn((RPW_tin(4)-RPW_tin(3)),-12) ~= roundn((Rtr_tin(4)-Rtr_tin(3)),-12)
fprintf('2st Reset after Forming ,step for tr and PW Is NOT the same');
end

Rtf_NumDataPts=Rtr_NumDataPts; %Forming Fall Time details
Rtf_StartVin=Rtr_StopVin;
Rtf_StopVin=Rtr_StartVin;
Rtf_Starttin=Rtf_Stopitin+(Ftr_tin(2)-Ftr_tin(1));
Rtf_Stopitin=Rtf_Starttin+Tr_Tf_TimeTotal;
Rtf_Vin=linspace(Rtf_StartVin,Rtf_StopVin,Rtf_NumDataPts+1);
Rtf_tin=linspace(Rtf_Starttin,Rtf_Stopitin,Rtf_NumDataPts+1);

R1Vin=[Rtr_Vin RPW_Vin Rtf_Vin];
R1tin=[Rtr_tin RPW_tin Rtf_tin];

Vin=[Vin R1Vin];
tin=[tin R1tin];

figure;
plot(R1tin,R1Vin);
TotTimeRelaxReset3=tin(end);

Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];
%---------------------------------------------------------------------------

Set3

Str_NumDataPts=Ftr_NumDataPts; %Forming Rise Time details
Str_StartVin=0;
Str_StopVin=SetPH;
Str_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Str_Stopitin=Str_Starttin+Tr_Tf_TimeTotal;
Str_Vin=linspace(Str_StartVin,Str_StopVin,Str_NumDataPts+1);
Str_tin=linspace(Str_Starttin,Str_Stoptin,Str_NumDataPts+1);

SPW_NumDataPts=1000;  %Forming Rise Time details
SPW_StartVin=SetPH;
SPW_StopVin=SetPH;
SPW_Starttin=Str_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
SPW_Stopoptin=SPW_Starttin+SetPW;
SPW_Vin=linspace(SPW_StartVin,SPW_StopVin,SPW_NumDataPts+1);
SPW_tin=linspace(SPW_Starttin,SPW_Stopoptin,SPW_NumDataPts+1);
if roundn((SPW_tin(4)-SPW_tin(3)),-12) ~= roundn((Str_tin(4)-Str_tin(3)),-12)
    fprintf('1st Reset after Forming ,step for tr and PW is NOT the same');
end

Stf_NumDataPts=Str_NumDataPts;  %Forming Fall Time details
Stf_StartVin=Str_STOPVin;
Stf_StopVin=Str_StartVin;
Stf_Starttin=SPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Stf_Stoptin=Stf_Starttin+Tr_Tf_TimeTotal;
Stf_Vin=linspace(Stf_StartVin,Stf_StopVin,Stf_NumDataPts+1);
Stf_tin=linspace(Stf_Starttin,Stf_Stoptin,Stf_NumDataPts+1);
S1Vin=[Str_Vin SPW_Vin Stf_Vin];
S1tin=[Str_tin SPW_tin Stf_tin];

Vin=[Vin S1Vin];
tin=[tin S1tin];

% figure;
% plot(S1tin,S1Vin);
TotTimeRelaxSet3=tin(end);
%--------------------------------------------------------------------------
Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];
%--------------------------------------------------------------------------
%4rd Reset
Rtr_NumDataPts=Ftr_NumDataPts;  %Forming Rise Time details
Rtr_StartVin=0;
Rtr_StopVin=ResetPH;
Rtr_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Rtr_Stopoptin=Rtr_Starttin+Tr_Tf_TimeTotal;
Rtr_Vin=linspace(Rtr_StartVin,Rtr_StopVin,Rtr_NumDataPts+1);
Rtr_tin=linspace(Rtr_Starttin,Rtr_Stopoptin,Rtr_NumDataPts+1);
RPW_NumDataPts=1000;  %Forming Rise Time details
RPW_StartVin=ResetPH;
RPW_StopVin=ResetPH;
RPW_Starttin=Rtr_Stopoptin+(Ftr_tin(2)-Ftr_tin(1));
RPW_Stopoptin=RPW_Starttin+ResetPW;
RPW_Vin=linspace(RPW_StartVin,RPW_StopVin,RPW_NumDataPts+1);
RPW_tin=linspace(RPW_Starttin,RPW_Stopoptin,RPW_NumDataPts+1);
if roundn((RPW_tin(4)-RPW_tin(3)),-12) ~= roundn((Rtr_tin(4)-Rtr_tin(3)),-12)
    fprintf('2st Reset after Forming ,step for tr and PW is NOT the same');
end

Rtf_NumDataPts=Rtr_NumDataPts;  %Forming Fall Time details
Rtf_StartVin=Rtr_StopVin;
Rtf_StopVin=Rtr_StartVin;
Rtf_Starttin=RPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Rtf_Stoptin=Rtf_Starttin+tr_Tf_TimeTotal;
Rtf_Vin=linspace(Rtf_StartVin,Rtf_StopVin,Rtf_NumDataPts+1);
Rtf_tin=linspace(Rtf_Starttin,Rtf_Stoptin,Rtf_NumDataPts+1);
R1Vin=[Rtr_Vin RPW_Vin Rtf_Vin];
R1tin=[Rtr_tin RPW_tin Rtf_tin];
Vin=[Vin R1Vin];
tin=[tin R1tin];

% figure;
% plot(R1tin,R1Vin);

TotTimeRelaxReset4=tin(end);

Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];

% Set4
Str_NumDataPts=Ftr_NumDataPts;  %Forming Rise Time details
Str_StartVin=0;
Str_StopVin=SetPH;
Str_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Str_Stoptin=Str_Starttin+tr_Tf_TimeTotal;
Str_Vin=linspace(Str_StartVin,Str_StopVin,Str_NumDataPts+1);
Str_tin=linspace(Str_Starttin,Str_Stoptin,Str_NumDataPts+1);

SPW_NumDataPts=1000;  %Forming Rise Time details
SPW_StartVin=SetPH;
SPW_StopVin=SetPH;
SPW_Starttin=Str_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
SPW_Stoptin=SPW_Starttin+SetPW;
SPW_Vin=linspace(SPW_StartVin,SPW_StopVin,SPW_NumDataPts+1);
SPW_tin=linspace(SPW_Starttin,SPW_Stoptin,SPW_NumDataPts+1);

if roundn((SPW_tin(4)-SPW_tin(3)),-12) ~= roundn((Str_tin(4)-Str_tin(3)),-12)
  fprintf('1st Reset after Forming ,step for tr and PW is NOT the same');
end

Stf_NumDataPts=Str_NumDataPts; %Forming Fall Time details
Stf_StartVin=Str_StopVin;
Stf_StopVin=Str_StartVin;
Stf_Starttin=SPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Stf_Stoptin=Stf_Starttin+tr_Tf_TimeTotal;
Stf_Vin=linspace(Stf_StartVin,Stf_StopVin,Stf_NumDataPts+1);
Stf_tin=linspace(Stf_Starttin,Stf_Stoptin,Stf_NumDataPts+1);
S1Vin=[Str_Vin SPW_Vin Stf_Vin];
S1tin=[Str_tin SPW_tin Stf_tin];

Vin=[Vin S1Vin];
tin=[tin S1tin];

% figure;
% plot(S1tin,S1Vin);

TotTimeRelaxSet4=tin(end);

Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];

5rd Reset
\begin{verbatim}
Rtr_NumDataPts=Ftr_NumDataPts;  %Forming Rise Time details
Rtr_StartVin=0;
Rtr_StopVin=ResetPH;
Rtr_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Rtr_Stoptin=Rtr_Starttin+Tr_Tf_TimeTotal;
Rtr_Vin=linspace(Rtr_StartVin,Rtr_StopVin,Rtr_NumDataPts+1);
Rtr_tin=linspace(Rtr_Starttin,Rtr_Stoptin,Rtr_NumDataPts+1);

RPW_NumDataPts=1000;  %Forming Rise Time details
RPW_StartVin=ResetPH;
RPW_StopVin=ResetPH;
RPW_Starttin=Rtr_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
RPW_Stoptin=RPW_Starttin+ResetPW;
RPW_Vin=linspace(RPW_StartVin,RPW_StopVin,RPW_NumDataPts+1);
RPW_tin=linspace(RPW_Starttin,RPW_Stoptin,RPW_NumDataPts+1);
if roundn((RPW_tin(4)-RPW_tin(3)),-12) ~= roundn((Rtr_tin(4)-Rtr_tin(3)),-12)
fprintf('2st Reset after Forming ,step for tr and PW is NOT the same');
end

Rtf_NumDataPts=Rtr_NumDataPts; %Forming Fall Time details
Rtf_StartVin=Rtr_StopVin;
Rtf_StopVin=Rtr_StartVin;
Rtf_Starttin=RPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Rtf_Stoptin=Rtf_Starttin+Tr_Tf_TimeTotal;
Rtf_Vin=linspace(Rtf_StartVin,Rtf_StopVin,Rtf_NumDataPts+1);
Rtf_tin=linspace(Rtf_Starttin,Rtf_Stoptin,Rtf_NumDataPts+1);
R1Vin=[Rtr_Vin RPW_Vin Rtf_Vin];
R1tin=[Rtr_tin RPW_tin Rtf_tin];

Vin=[Vin R1Vin];
tin=[tin R1tin];

% figure;
% plot(R1tin,R1Vin);

TotTimeRelaxReset5=tin(end);

%-------------------------------------------------------------------------
Str_NumDataPts=Ftr_NumDataPts;  %Forming Rise Time details
Str_StartVin=0;
Str_StopVin=SetPH;
Str_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Str_Stoptin=Str_Starttin+Tr_Tf_TimeTotal;
Str_Vin=linspace(Str_StartVin,Str_StopVin,Str_NumDataPts+1);
Str_tin=linspace(Str_Starttin,Str_Stoptin,Str_NumDataPts+1);

SPW_NumDataPts=1000;  %Forming Rise Time details
SPW_StartVin=SetPH;
SPW_StopVin=SetPH;
SPW_Starttin=Str_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
SPW_Stoptin=SPW_Starttin+SetPW;
SPW_Vin=linspace(SPW_StartVin,SPW_StopVin,SPW_NumDataPts+1);
SPW_tin=linspace(SPW_Starttin,SPW_Stoptin,SPW_NumDataPts+1);
\end{verbatim}
if roundn((SPW_tin(4)-SPW_tin(3)),-12) ~= roundn((Str_tin(4)-Str_tin(3)),-12)
fprintf('1st Reset after Forming ,step for tr and PW is NOT the same');
end

Stf_NumDataPts=Str_NumDataPts; %Forming Fall Time details
Stf_StartVin=Str_StartVin;
Stf_StopVin=Str_StopVin;
Stf_Starttin=SPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Stf_Stoptin=Stf_Starttin+Tr_Tf_TimeTotal;
Stf_Vin=linspace(Stf_StartVin,Stf_StopVin,Stf_NumDataPts+1);
Stf_tin=linspace(Stf_Starttin,Stf_Stoptin,Stf_NumDataPts+1);

S1Vin=[Str_Vin SPW_Vin Stf_Vin];
S1tin=[Str_tin SPW_tin Stf_tin];

Vin=[Vin S1Vin];
tin=[tin S1tin];

% figure;
% plot(S1tin,S1Vin);
TotTimeRelaxSet5=tin(end);

%--------------------------------------------------------------------------
Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];
%--------------------------------------------------------------------------
%6rd Reset
Rtr_NumDataPts=Ftr_NumDataPts; %Forming Rise Time details
Rtr_StartVin=0;
Rtr_StopVin=ResetPH;
Rtr_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Rtr_Stoptin=Rtr_Starttin+Tr_Tf_TimeTotal;
Rtr_Vin=linspace(Rtr_StartVin,Rtr_StopVin,Rtr_NumDataPts+1);
Rtr_tin=linspace(Rtr_Starttin,Rtr_Stoptin,Rtr_NumDataPts+1);

RPW_NumDataPts=1000; %Forming Rise Time details
RPW_StartVin=ResetPH;
RPW_StopVin=ResetPH;
RPW_Starttin=Rtr_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
RPW_Stoptin=RPW_Starttin+ResetPW;
RPW_Vin=linspace(RPW_StartVin,RPW_StopVin,RPW_NumDataPts+1);
RPW_tin=linspace(RPW_Starttin,RPW_Stoptin,RPW_NumDataPts+1);

if roundn((RPW_tin(4)-RPW_tin(3)),-12) ~= roundn((Rtr_tin(4)-Rtr_tin(3)),-12)
fprintf('2st Reset after Forming ,step for tr and PW is NOT the same');
end

Rtf_NumDataPts=Rtr_NumDataPts; %Forming Fall Time details
Rtf_StartVin=Rtr_StopVin;
Rtf_StopVin=Rtr_StartVin;
Rtf_Starttin=RPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Rtf_Stoptin=Rtf_Starttin+Tr_Tf_TimeTotal;
Rtf_Vin=linspace(Rtf_StartVin,Rtf_StopVin,Rtf_NumDataPts+1);
Rtf_tin=linspace(Rtf_Starttin,Rtf_Stoptin,Rtf_NumDataPts+1);

R1Vin=[Rtr_Vin RPW_Vin Rtf_Vin];
R1tin=[Rtr_tin RPW_tin Rtf_tin];

Vin=[Vin R1Vin];
tin=[tin R1tin];

% figure;
```matlab
% plot(R1tin,R1Vin);  
TotTimeRelaxReset6=tin(end);  
———————————————————————————————————
Vin=[Vin 0];  
tin=[tin tin(end)+RelaxationTime];  
———————————————————————————————————

% Set
Str_NumDataPts=Ftr_NumDataPts;  %Forming Rise Time details
Str_StartVin=0;
Str_StopVin=SetPH;
Str_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Str_Stopoptin=Str_Starttin+Tr_Tf_TimeTotal;
Str_Vin=linspace(Str_StartVin,Str_StopVin,Str_NumDataPts+1);  
Str_tin=linspace(Str_Starttin,Str_Stopoptin,Str_NumDataPts+1);  

SPW_NumDataPts=1000;  %Forming Rise Time details
SPW_StartVin=SetPH;
SPW_StopVin=SetPH;
SPW_Starttin=Str_Stopoptin+(Ftr_tin(2)-Ftr_tin(1));
SPW_Stopoptin=SPW_Starttin+SetPW;
SPW_Vin=linspace(SPW_StartVin,SPW_StopVin,SPW_NumDataPts+1);  
SPW_tin=linspace(SPW_Starttin,SPW_Stopoptin,SPW_NumDataPts+1);
if roundn((SPW_tin(4)-SPW_tin(3)),-12) ~= roundn((Str_tin(4)-Str_tin(3)),-12)
    fprintf('1st Reset after Forming ,step for tr and PW is NOT the same');
end

Stf_NumDataPts=Str_NumDataPts; %Forming Fall Time details
Stf_StartVin=Str_StopVin;
Stf_StopVin=Str_StartVin;
Stf_Starttin=SPW_Stopoptin+(Ftr_tin(2)-Ftr_tin(1));
Stf_Stopoptin=Stf_Starttin+Tr_Tf_TimeTotal;
Stf_Vin=linspace(Stf_StartVin,Stf_StopVin,Stf_NumDataPts+1);  
Stf_tin=linspace(Stf_Starttin,Stf_Stopoptin,Stf_NumDataPts+1);  

S1Vin=[Str_Vin SPW_Vin Stf_Vin];
S1tin=[Str_tin SPW_tin Stf_tin];

Vin=[Vin S1Vin];
tin=[tin S1tin];

% figure;
% plot(S1tin,S1Vin);
TotTimeRelaxSet6=tin(end);  
———————————————————————————————————
Vin=[Vin 0];  
tin=[tin tin(end)+RelaxationTime];  
———————————————————————————————————

% 7rd Reset
Rtr_NumDataPts=Ftr_NumDataPts;  %Forming Rise Time details
Rtr_StartVin=0;
Rtr_StopVin=ResetPH;
Rtr_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Rtr_Stopoptin=Rtr_Starttin+Tr_Tf_TimeTotal;
Rtr_Vin=linspace(Rtr_StartVin,Rtr_StopVin,Rtr_NumDataPts+1);  
Rtr_tin=linspace(Rtr_Starttin,Rtr_Stopoptin,Rtr_NumDataPts+1);  

RPW_NumDataPts=1000;  %Forming Rise Time details
```

The image contains a page of MATLAB code. The code appears to be related to signal processing or circuit simulation, involving the plotting of signals, setting of time intervals, and the calculation of specific times related to relaxation and reset processes. The code includes multiple sections for setting different signal characteristics and times, which are aggregated into various variables for further plotting and analysis.
RPW_StartVin=ResetPH;
RPW_StopVin=ResetPH;
RPW_Starttin=Rtr_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
RPW_Stoptin=RPW_Starttin+ResetPW;
RPW_Vin=linspace(RPW_StartVin,RPW_StopVin,RPW_NumDataPts+1);
RPW_tin=linspace(RPW_Starttin,RPW_Stoptin,RPW_NumDataPts+1);

if roundn((RPW_tin(4)-RPW_tin(3)),-12) ~= roundn((Rtr_tin(4)-Rtr_tin(3)),-12)
fprintf('2st Reset after Forming , step for tr and PW is NOT the same');
end

Rtf_NumDataPts=Rtr_NumDataPts; % Forming Fall Time details
Rtf_StartVin=Rtr_StopVin;
Rtf_StopVin=Rtr_StartVin;
Rtf_Starttin=RPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Rtf_Stoptin=Rtf_Starttin+Tr_Tf_TimeTotal;
Rtf_Vin=linspace(Rtf_StartVin,Rtf_StopVin,Rtf_NumDataPts+1);
Rtf_tin=linspace(Rtf_Starttin,Rtf_Stoptin,Rtf_NumDataPts+1);

R1Vin=[Rtr_Vin RPW_Vin Rtf_Vin];
R1tin=[Rtr_tin RPW_tin Rtf_tin];

Vin=[Vin R1Vin];
tin=[tin R1tin];

% figure;
% plot(R1tin,R1Vin);

TotTimeRelaxReset7=tin(end);

Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];

% Set7
Str_NumDataPts=Ftr_NumDataPts; % Forming Rise Time details
Str_StartVin=0;
Str_StopVin=SetPH;
Str_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Str_Stoptin=Str_Starttin+Tr_Tf_TimeTotal;
Str_Vin=linspace(Str_StartVin,Str_StopVin,Str_NumDataPts+1);
Str_tin=linspace(Str_Starttin,Str_Stoptin,Str_NumDataPts+1);

SPW_NumDataPts=1000; % Forming Rise Time details
SPW_StartVin=SetPH;
SPW_StopVin=SetPH;
SPW_Starttin=Str_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
SPW_Stoptin=SPW_Starttin+SetPW;
SPW_Vin=linspace(SPW_StartVin,SPW_StopVin,SPW_NumDataPts+1);
SPW_tin=linspace(SPW_Starttin,SPW_Stoptin,SPW_NumDataPts+1);

if roundn((SPW_tin(4)-SPW_tin(3)),-12) ~= roundn((Str_tin(4)-Str_tin(3)),-12)
fprintf('1st Reset after Forming , step for tr and PW is NOT the same');
end

Stf_NumDataPts=Str_NumDataPts; % Forming Fall Time details
Stf_StartVin=Str_StopVin;
Stf_Starttin=Str_StartVin;
Stf_Stoptin=SPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Stf_Stoptin=Stf_Starttin+Tr_Tf_TimeTotal;
Stf_Vin=linspace(Stf_StartVin,Stf_StopVin,Stf_NumDataPts+1);
Stf_tin=linspace(Stf_Starttin,Stf_Stoptin,Stf_NumDataPts+1);
S1Vin=[Str_Vin SPW_Vin Stf_Vin];
S1tin=[Str_tin SPW_tin Stf_tin]

Vin=[Vin S1Vin];
tin=[tin S1tin];

% figure;
% plot(S1tin,S1Vin);
TotTimeRelaxSet7=tin(end);

%---------------------------------------------------------------------
Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];

%---------------------------------------------------------------------

% 8rd Reset

Rtr_NumDataPts=Ftr_NumDataPts;  %Forming Rise Time details
Rtr_StartVin=0;
Rtr_StopVin=ResetPH;
Rtr_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Rtr_Stopoptin=Rtr_Starttin+Tr_Tf_TimeTotal;
Rtr_Vin=linspace(Rtr_StartVin,Rtr_StopVin,Rtr_NumDataPts+1);
Rtr_tin=linspace(Rtr_Starttin,Rtr_Stopoptin,Rtr_NumDataPts+1);

RPW_NumDataPts=1000;  %Forming Rise Time details
RPW_StartVin=ResetPH;
RPW_StopVin=ResetPH;
RPW_Starttin=Rtr_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
RPW_Stopoptin=RPW_Startoptin+ResetPW;
RPW_Vin=linspace(RPW_StartVin,RPW_StopVin,RPW_NumDataPts+1);
RPW_tin=linspace(RPW_Starttin,RPW_Stopoptin,RPW_NumDataPts+1);

if roundn((RPW_tin(4)-RPW_tin(3)),-12) ~= roundn((Rtr_tin(4)-Rtr_tin(3)),-12)
fprintf('2st Reset after Forming ,step for tr and PW is NOT the same');
end

Rtf_NumDataPts=Rtr_NumDataPts; %Forming Fall Time details
Rtf_StartVin=Rtr_StopVin;
Rtf_StopVin=Rtr_StartVin;
Rtf_Starttin=RPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Rtf_Stopoptin=Rtf_Startoptin+Tr_Tf_TimeTotal;
Rtf_Vin=linspace(Rtf_StartVin,Rtf_StopVin,Rtf_NumDataPts+1);
Rtf_tin=linspace(Rtf_Starttin,Rtf_Stopoptin,Rtf_NumDataPts+1);

R1Vin=[Rtr_Vin RPW_Vin Rtf_Vin];
R1tin=[Rtr_tin RPW_tin Rtf_tin];

Vin=[Vin R1Vin];
tin=[tin R1tin];

% figure;
% plot(R1tin,R1Vin);

TotTimeRelaxReset8=tin(end);

%---------------------------------------------------------------------
Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];

%---------------------------------------------------------------------

% Set8
Str_NumDataPts=Ftr_NumDataPts;  %Forming Rise Time details
Str_StartVin=0;
Str_StopVin=SetPH;
Str_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Str_Stoptin=Str_Starttin+Tr_Tf_TimeTotal;
Str_Vin=linspace(Str_StartVin,Str_StopVin,Str_NumDataPts+1);
Str_tin=linspace(Str_Starttin,Str_Stoptin,Str_NumDataPts+1);

SPW_NumDataPts=1000;  %Forming Rise Time details
SPW_StartVin=SetPH;
SPW_StopVin=SetPH;
SPW_Starttin=Str_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
SPW_Stoptin=SPW_Starttin+SetPW;
SPW_Vin=linspace(SPW_StartVin,SPW_StopVin,SPW_NumDataPts+1);
SPW_tin=linspace(SPW_Starttin,SPW_Stoptin,SPW_NumDataPts+1);

if roundn((SPW_tin(4)-SPW_tin(3)),-12) ~= roundn((Str_tin(4)-Str_tin(3)),-12)
    fprintf('1st Reset after Forming ,step for tr and PW is NOT the same');
end

Stf_NumDataPts=Str_NumDataPts; %Forming Fall Time details
Stf_StartVin=Str_StopVin;
Stf_StopVin=Str_StartVin;
Stf_Starttin=SPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Stf_Stoptin=Stf_Starttin+Tr_Tf_TimeTotal;
Stf_Vin=linspace(Stf_StartVin,Stf_StopVin,Stf_NumDataPts+1);
Stf_tin=linspace(Stf_Starttin,Stf_Stoptin,Stf_NumDataPts+1);

S1Vin=[Str_Vin SPW_Vin Stf_Vin];
S1tin=[Str_tin SPW_tin Stf_tin];

Vin=[Vin S1Vin];
tin=[tin S1tin];

if roundn((RPW_tin(4)-RPW_tin(3)),-12) ~= roundn((Rtr_tin(4)-Rtr_tin(3)),-12)
    fprintf('2st Reset after Forming ,step for tr and PW is NOT the same');
end
Rtf_NumDataPts=Rtr_NumDataPts; %Forming Fall Time details
Rtf_StartVin=Rtr_StopVin;
Rtf_StopVin=Rtr_StartVin;
Rtf_Starttin=RPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Rtf_Stoptin=Rtf_Starttin+Tr_Tf_TimeTotal;
Rtf_Vin=linspace(Rtf_StartVin,Rtf_StopVin,Rtf_NumDataPts+1);
Rtf_tin=linspace(Rtf_Starttin,Rtf_Stoptin,Rtf_NumDataPts+1);

R1Vin=[Rtr_Vin RPW_Vin Rtf_Vin];
R1tin=[Rtr_tin RPW_tin Rtf_tin];

Vin=[Vin R1Vin];
tin=[tin R1tin];

% figure;
% plot(R1tin,R1Vin);

TotTimeRelaxReset9=tin(end);
%--------------------------------------------------------------------------
Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];
%--------------------------------------------------------------------------
%-------------------------------------------------------------------------
%Set9
Str_NumDataPts=Ftr_NumDataPts; %Forming Rise Time details
Str_StartVin=0;
Str_StopVin=SetPH;
Str_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Str_Stoptin=Str_Starttin+Tr_Tf_TimeTotal;
Str_Vin=linspace(Str_StartVin,Str_StopVin,Str_NumDataPts+1);
Str_tin=linspace(Str_Starttin,Str_Stoptin,Str_NumDataPts+1);

SPW_NumDataPts=1000; %Forming Rise Time details
SPW_StartVin=SetPH;
SPW_StopVin=SetPH;
SPW_Starttin=Str_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
SPW_Stoptin=SPW_Starttin+SetPW;
SPW_Vin=linspace(SPW_StartVin,SPW_StopVin,SPW_NumDataPts+1);
SPW_tin=linspace(SPW_Starttin,SPW_Stoptin,SPW_NumDataPts+1);

if roundn((SPW_tin(4)-SPW_tin(3)),-12) ~= roundn((Str_tin(4)-Str_tin(3)),-12)
    fprintf('1st Reset after Forming ,step for tr and PW is NOT the same');
end

Stf_NumDataPts=Str_NumDataPts; %Forming Fall Time details
Stf_StartVin=Str_StopVin;
Stf_StopVin=Str_StartVin;
Stf_Starttin=SPW_Stoptin+(Ftr_tin(2)-Ftr_tin(1));
Stf_Stoptin=Stf_Starttin+Tr_Tf_TimeTotal;
Stf_Vin=linspace(Stf_StartVin,Stf_StopVin,Stf_NumDataPts+1);
Stf_tin=linspace(Stf_Starttin,Stf_Stoptin,Stf_NumDataPts+1);

S1Vin=[Str_Vin SPW_Vin Stf_Vin];
S1tin=[Str_tin SPW_tin Stf_tin];

Vin=[Vin S1Vin];
tin=[tin S1tin];

% figure;
% plot(S1tin,S1Vin);

TotTimeRelaxSet9=tin(end);
Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];
%--------------------------------------------------------------------------
%10rd Reset

Rtr_NumDataPts=Ftr_NumDataPts;  %Forming Rise Time details
Rtr_StartVin=0;
Rtr_StopVin=ResetPH;
Rtr_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Rtr_Stopoptin=Rtr_Starttin+Tr_Tf_TimeTotal;
Rtr_Vin=linspace(Rtr_StartVin,Rtr_StopVin,Rtr_NumDataPts+1);
Rtr_tin=linspace(Rtr_Starttin,Rtr_Stopoptin,Rtr_NumDataPts+1);

RPW_NumDataPts=1000;  %Forming Rise Time details
RPW_StartVin=ResetPH;
RPW_StopVin=ResetPH;
RPW_Starttin=Rtr_Stopoptin+(Ftr_tin(2)-Ftr_tin(1));
RPW_Stopoptin=RPW_Starttin+ResetPW;
RPW_Vin=linspace(RPW_StartVin,RPW_StopVin,RPW_NumDataPts+1);
RPW_tin=linspace(RPW_Starttin,RPW_Stopoptin,RPW_NumDataPts+1);

if roundn((RPW_tin(4)-RPW_tin(3)),-12) ~= roundn((Rtr_tin(4)-Rtr_tin(3)),-12)
fprintf('2st Reset after Forming ,step for tr and PW is NOT the same');
end

Rtf_NumDataPts=Rtr_NumDataPts; %Forming Fall Time details
Rtf_StartVin=Rtr_StopVin;
Rtf_StopVin=Rtr_StartVin;
Rtf_Starttin=RPW_Stopoptin+(Ftr_tin(2)-Ftr_tin(1));
Rtf_Stopoptin=Rtf_Starttin+Tr_Tf_TimeTotal;
Rtf_Vin=linspace(Rtf_StartVin,Rtf_StopVin,Rtf_NumDataPts+1);
Rtf_tin=linspace(Rtf_Starttin,Rtf_Stopoptin,Rtf_NumDataPts+1);

R1Vin=[Rtr_Vin RPW_Vin Rtf_Vin];
R1tin=[Rtr_tin RPW_tin Rtf_tin];

Vin=[Vin R1Vin];
tin=[tin R1tin];

% figure;
% plot(R1tin,R1Vin);

TotTimeRelaxReset10=tin(end);
%--------------------------------------------------------------------------

Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];
%--------------------------------------------------------------------------

%Set10

Str_NumDataPts=Ftr_NumDataPts;  %Forming Rise Time details
Str_StartVin=0;
Str_StopVin=SetPH;
Str_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Str_Stopoptin=Str_Starttin+Tr_Tf_TimeTotal;
Str_Vin=linspace(Str_StartVin,Str_StopVin,Str_NumDataPts+1);
Str_tin=linspace(Str_Starttin,Str_Stopoptin,Str_NumDataPts+1);

SPW_NumDataPts=1000;  %Forming Rise Time details
SPW_StartVin=SetPH;
SPW_StopVin=SetPH;
SPW_Starttin=Str_Stopoptin+(Ftr_tin(2)-Ftr_tin(1));
SPW_Stoptin = SPW_Starttin + SPW_SetPW;
SPW_Vin = linspace(SPW_StartVin, SPW_StopVin, SPW_NumDataPts + 1);
SPW_tin = linspace(SPW_Starttin, SPW_Stoptin, SPW_NumDataPts + 1);

if roundn((SPW_tin(4) - SPW_tin(3)), -12) ~= roundn((Str_tin(4) - Str_tin(3)), -12)
fprintf('1st Reset after Forming, step for tr and PW is NOT the same');
end

Stf_NumDataPts = Str_NumDataPts; % Forming Fall Time details
Stf_StartVin = Str_StopVin;
Stf_StopVin = Str_StartVin;
Stf_Starttin = SPW_Stoptin + (Ftr_tin(2) - Ftr_tin(1));
Stf_Stoptin = Stf_Starttin + Tr_Tf_TimeTotal;
Stf_Vin = linspace(Stf_StartVin, Stf_StopVin, Stf_NumDataPts + 1);
Stf_tin = linspace(Stf_Starttin, Stf_Stoptin, Stf_NumDataPts + 1);

S1Vin = [Str_Vin SPW_Vin Stf_Vin];
S1tin = [Str_tin SPW_tin Stf_tin];

Vin = [Vin S1Vin];
tin = [tin S1tin];

% figure;
% plot(S1tin, S1Vin);
TotTimeRelaxSet10 = tin(end);
%------------------------------------------------------------------------
Vin = [Vin 0];
tin = [tin tin(end) + RelaxationTime];
%------------------------------------------------------------------------

11rd Reset

Rtr_NumDataPts = Ftr_NumDataPts; % Forming Rise Time details
Rtr_StartVin = 0;
Rtr_StopVin = ResetPH;
Rtr_Starttin = tin(end) + (Ftr_tin(2) - Ftr_tin(1));
Rtr_Stoptin = Rtr_Starttin + Tr_Tf_TimeTotal;
Rtr_Vin = linspace(Rtr_StartVin, Rtr_StopVin, Rtr_NumDataPts + 1);
Rtr_tin = linspace(Rtr_Starttin, Rtr_Stoptin, Rtr_NumDataPts + 1);

RPW_NumDataPts = 1000; % Forming Rise Time details
RPW_StartVin = ResetPH;
RPW_StopVin = ResetPH;
RPW_Starttin = Rtr_Starttin + (Ftr_tin(2) - Ftr_tin(1));
RPW_Stoptin = RPW_Starttin + ResetPW;
RPW_Vin = linspace(RPW_StartVin, RPW_StopVin, RPW_NumDataPts + 1);
RPW_tin = linspace(RPW_Starttin, RPW_Stoptin, RPW_NumDataPts + 1);

if roundn((RPW_tin(4) - RPW_tin(3)), -12) ~= roundn((Rtr_tin(4) - Rtr_tin(3)), -12)
fprintf('2st Reset after Forming, step for tr and PW is NOT the same');
end

Rtf_NumDataPts = Rtr_NumDataPts; % Forming Fall Time details
Rtf_StartVin = Rtr_StopVin;
Rtf_StopVin = Rtr_StartVin;
Rtf_Starttin = RPW_Stoptin + (Ftr_tin(2) - Ftr_tin(1));
Rtf_Stoptin = Rtf_Starttin + Tr_Tf_TimeTotal;
Rtf_Vin = linspace(Rtf_StartVin, Rtf_StopVin, Rtf_NumDataPts + 1);
Rtf_tin = linspace(Rtf_Starttin, Rtf_Stoptin, Rtf_NumDataPts + 1);

R1Vin = [Rtr_Vin RPW_Vin Rtf_Vin];
R1tin = [Rtr_tin RPW_tin Rtf_tin];
Vin=[Vin R1Vin];
tin=[tin R1tin];

% figure;
% plot(R1tin,R1Vin);

TotTimeRelaxReset11=tin(end);

Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];

% Set11
Str_NumDataPts=Ftr_NumDataPts;  %Forming Rise Time details
Str_StartVin=0;
Str_StopVin=SetPH;
Str_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Str_Stopoptin=Str_Starttin+Tr_Tf_TimeTotal;
Str_Vin=linspace(Str_StartVin,Str_StopVin,Str_NumDataPts+1);
Str_tin=linspace(Str_Starttin,Str_Stopoptin,Str_NumDataPts+1);

SPW_NumDataPts=1000;  %Forming Rise Time details
SPW_StartVin=SetPH;
SPW_StopVin=SetPH;
SPW_Starttin=Str_Stopoptin+(Ftr_tin(2)-Ftr_tin(1));
SPW_Stopoptin=SPW_Starttin+SetPW;
SPW_Vin=linspace(SPW_StartVin,SPW_StopVin,SPW_NumDataPts+1);
SPW_tin=linspace(SPW_Starttin,SPW_Stopoptin,SPW_NumDataPts+1);

if roundn((SPW_tin(4)-SPW_tin(3)),-12) ~= roundn((Str_tin(4)-Str_tin(3)),-12)
    fprintf('1st Reset after Forming ,step for tr and PW is NOT the same');
end

Stf_NumDataPts=Str_NumDataPts; %Forming Fall Time details
Stf_StartVin=Str_StopVin;
Stf_StopVin=Str_StartVin;
Stf_Starttin=SPW_Stopoptin+(Ftr_tin(2)-Ftr_tin(1));
Stf_Stopoptin=Stf_Starttin+Tr_Tf_TimeTotal;
Stf_Vin=linspace(Stf_StartVin,Stf_StopVin,Stf_NumDataPts+1);
Stf_tin=linspace(Stf_Starttin,Stf_Stopoptin,Stf_NumDataPts+1);

S1Vin=[Str_Vin SPW_Vin Stf_Vin];
S1tin=[Str_tin SPW_tin Stf_tin];

Vin=[Vin S1Vin];
tin=[tin S1tin];

% figure;
% plot(S1tin,S1Vin);

TotTimeRelaxSet11=tin(end);

Vin=[Vin 0];
tin=[tin tin(end)+RelaxationTime];

% Set11

% 12rd Reset
Rtr_NumDataPts=Ftr_NumDataPts;  %Forming Rise Time details
Rtr_StartVin=0;
Rtr_StopVin=ResetPH;
Rtr_Starttin=tin(end)+(Ftr_tin(2)-Ftr_tin(1));
Rtr_Stopoptin=Rtr_Starttin+Tr_Tf_TimeTotal;
Rtr_Vin=linspace(Rtr_StartVin,Rtr_StopVin,Rtr_NumDataPts+1);
\begin{verbatim}
Rtr_tin = linspace(Rtr_Starttin, Rtr_Stoptin, Rtr_NumDataPts+1);

RFW_NumDataPts = 1000;  % Forming Rise Time details
RFW_StartVin = ResetPH;
RFW_StopVin = ResetPH;
RFW_Starttin = Rtr_Stoptin + (Ftr_tin(2) - Ftr_tin(1));
RFW_Stopin = RFW_Starttin + ResetPW;
RFW_Vin = linspace(RFW_StartVin, RFW_StopVin, RFW_NumDataPts+1);
RFW_tin = linspace(RFW_Starttin, RFW_Stopin, RFW_NumDataPts+1);
if roundn((RFW_tin(4)-RFW_tin(3)),-12) ~= roundn((Rtr_tin(4)-Rtr_tin(3)),-12)
    fprintf('2st Reset after Forming , step for tr and PW is NOT the same');
end

Rtf_NumDataPts = Rtr_NumDataPts;  % Forming Fall Time details
Rtf_StartVin = Rtr_StopVin;
Rtf_StopVin = Rtr_StartVin;
Rtf_Starttin = RFW_Stoptin + (Ftr_tin(2) - Ftr_tin(1));
Rtf_Stopin = Rtf_Starttin + Tr_Tf_TimeTotal;
Rtf_Vin = linspace(Rtf_StartVin, Rtf_StopVin, Rtf_NumDataPts+1);
Rtf_tin = linspace(Rtf_Starttin, Rtf_Stopin, Rtf_NumDataPts+1);
R1Vin = [Rtr_Vin RFW_Vin Rtf_Vin];
R1tin = [Rtr_tin RFW_tin Rtf_tin];
Vin = [Vin R1Vin];
tin = [tin R1tin];

% figure;
% plot(R1tin, R1Vin);

TotTimeRelaxReset12 = tin(end);

Vin = [Vin 0];
tin = [tin tin(end)+RelaxationTime];

TotalVin = Vin;
Totaltin = tin;
end

function [ x, ro, Xmesh, Ymesh, Zmesh ] = nmap(vac_map, Xmesh, Ymesh, Zmesh)
% NMAP Calculation of defect density map (cm^-3)
% global layers;
% keyboard;

N = 1;
NN = 28;  % multiplications factor for 8 2.5A cubes --> .5A spacing
NNN = 1;
rt = lambda^2;
Lx = (xspacing(NNN+1)-xspacing(1))/1; Ly = (yspacing(NNN+1)-yspacing(1))/1; Lz = (zspacing(NNN+1)-zspacing(1))/1;
\end{verbatim}
[Nvac,~]=size(vac_map);
ro=zeros(size(Xmesh));
x=zeros(size(Xmesh));
mol_dens=zeros(size(Xmesh));
mol_dens(Zmesh>=0&Zmesh<=layers(1).t)=layers(1).mol_dens;
mol_dens(Zmesh>layers(1).t&Zmesh<=layers(1).t+layers(2).t)=layers(2).mol_dens;
for n=1:Nvac
    %finding density of vac with respect to Xmesh grid point, N defines the
    %interval of Xmesh spacing
    ro_def=normpdf_3D(Xmesh(1:N:end),vac_map(n,1),rt,Ymesh(1:N:end),vac_map(n,2),rt,
    Zmesh(1:N:end),vac_map(n,3),rt); %(1/m^3)
    % expands ro_def into 2.5A spacing
    % keyboard;
    %     ro_def=ro_def';
    %     ro_def=ro_def(ones(1,N),:);
    %     ro_def=ro_def(:)';
    %     ro_def=ro_def';
    C=sum(ro_def(:))*Lx*Ly*Lz;
    ro_def=ro_def./C;
    % if length(ro_def(:))~=length(Xmesh)
    %     flag=1
    %     ro_def=ro_def(1:length(Xmesh));
    %     ro_def=[ro_def(:); ro_def(end)];
    % end
    ro_def=reshape(ro_def(:),size(Xmesh));
ro=ro+ro_def;
end
x(Zmesh>=0&Zmesh<=layers(1).t+layers(2).t)=NN.*ro(Zmesh>=0&Zmesh<=layers(1).t+layers(2).t)./(mol_dens(Zmesh>=0&Zmesh<=layers(1).t+layers(2).t).*1e6);
% keyboard;
end

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

function [V T I P Jinf Jsup] = DC_Sim_Drift_T(xspacing,yspacing,zspacing,Vinf,Vsup,Tinf,Tsup,rox,roy,roz,kthx,kthy,kthz,Rinf,Rsup,Rthinf,Rthsup,alpha_T,V0,T0)
% DC_SIM_DRIFT_T Summary of this function goes here
% Detailed explanation goes here

% MESHGRID FORMAT IS ADOPTED!!!!!!
Xc=(xspacing(1:end-1)+xspacing(2:end))/2;
Yc=(yspacing(1:end-1)+yspacing(2:end))/2;
Zc=(zspacing(1:end-1)+zspacing(2:end))/2;
[xmesh,ymesh,zmesh]=meshgrid(xspacing,yspacing,zspacing);
[X1 Y1 Z1]=meshgrid(Xc,yspacing,zspacing);
[X2 Y2 Z2]=meshgrid(xspacing,Yc,zspacing);
[X3 Y3 Z3]=meshgrid(xspacing,yspacing,Zc);

switch nargin
    case 18
        % keyboard;
        V=zeros(length(yspacing),length(xspacing),length(zspacing));
T = Tinf*ones(length(yspacing), length(xspacing), length(zspacing));
case 20
    V = V0;
    T = T0;
end

varMax = 1e-2;
nstepmax = 50;
stop = 0;
i = 1;
while ~stop

   rox_ = rox_.*(1+alpha_T.*(interp3_linear(xmesh, ymesh, zmesh, T, X1, Y1, Z1, NaN) - 300));
    roy_ = roy_.*(1+alpha_T.*(interp3_linear(xmesh, ymesh, zmesh, T, X2, Y2, Z2, NaN) - 300));
    roz_ = roz_.*(1+alpha_T.*(interp3_linear(xmesh, ymesh, zmesh, T, X3, Y3, Z3, NaN) - 300));

    [V_new I P Jinf Jsup] = potmapCalc_drift(xspacing, yspacing, zspacing, Vinf, Vsup, Rinf, Rsup, rox_, roy_, roz_, V);
    fprintf('P1=%d, P2=%d
', trapz(xspacing, trapz(yspacing, trapz(zspacing, P, 3), 1), 2), I*(Vsup-Vinf));

    [T_new] = TmapCalc(xspacing, yspacing, zspacing, Tinf, Tsup, Rthinf, Rthsup, kthx, kthy, kthz, P, T);

    var1 = min(abs(V_new(:)-V(:)))/min(abs(V(:)))+varMax;
    var2 = max(abs(V_new(:)-V(:)))/max(abs(V(:)))+varMax;

    T = T_new;
    V = V_new;

    if var1<varMax & var2<varMax
        fprintf('convergence reached\n');
        stop = 1;
    end
    if i==nstepmax
        fprintf('maximum number of iterations reached\n');
        stop = 1;
    end
    fprintf(nsteps=%d, var1=%d, var2=%d, I=%d\n', i, var1, var2, I);
    i = i+1;
end

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

% FUNCTION USED INSTEAD OF INTERP3 TO SPEED UP!
% Calls the mex function ba_interp3: much faster!!

function F = interp3_linear(X,Y,Z,V,XI,YI,ZI,ExtrapVal)
F = ba_interp3(X,Y,Z,V,XI,YI,ZI, 'linear'); % c code
A=XI<X(1)|XI>X(end)|YI<Y(1)|YI>Y(end)|ZI<Z(1)|ZI>Z(end);
if numel(ExtrapVal) ==1
    F(A)=ExtrapVal;
else
    F(A)=ExtrapVal(A);
end
% Solution of drift equation over the domain defined by
% xspacing,yspacing,zspacing
% Input arguments:
% xspacing: 1xNx array
% yspacing: 1xNy array
% zspacing: 1xNz array
% Vinf: voltage at the bottom electrode
% Vsup: voltage at the top electrode
% Rinf: bottom electrode resistance
% Rsup: top electrode resistance
% ro: Nx x Ny x Nz resistivity map (ohm/m)
% V0: starting potential (used by numerical method)
function [V I P Jinf Jsup Jcenter]=potmapCalc_drift(xspacing,yspacing,zspacing,Vinf,Vsup,Rinf,Rsup,roX,roY,roZ,V0)
    Lx=xspacing(2)-xspacing(1);
    Ly=yspacing(2)-yspacing(1);
    Lz=zspacing(2)-zspacing(1);
    Rx=roX.*Lx./(Ly*Lz);
    Ry=roY.*Ly./(Lx*Lz);
    Rz=roZ.*(Lz./(Lx*Ly));
    Imap=zeros(length(xspacing),length(yspacing),length(zspacing));
% Solution of the charge equation through SOR method
[V]=SOR_3D(Lx,Ly,Lz,V0,1./Rx,1./Ry,1./Rz,1/Rinf,1/Rsup,Vinf,Vsup,Imap);
    Jinf=(V(:,:,2)-V(:,:,1))./(Lx.*Ly.*Rz(:,:,1));
    Jsup=(V(:,:,end)-V(:,:,end-1))./(Lx.*Ly.*Rz(:,:,1));
    Jcenter=(V(:,:,floor(end/2))-V(:,:,floor(end/2)-1))./(Lx.*Ly.*Rz(:,:,floor(end/2)-1));
    I=trapz(xspacing,trapz(yspacing,Jcenter,1),2);
    %I=sum(sum((V(:,:,2)-V(:,:,1))./Rz(:,:,1),2),1);
    P=zeros(length(yspacing),length(xspacing),length(zspacing));
% Power due to heat dissipation along x (Rx)
P(:,:,1:end-1,:)=P(:,:,1:end-1,:)+(V(:,:,1:end-1,:)-V(:,:,1:end-2,:)).^2./(2.*Rx);
P(:,:,2:end,:)=P(:,:,2:end,:)+\(V(:,:,2:end,:)-V(:,:,1:end-1,:))\.^2./(2.*Rx);
% Power due to heat dissipation along y (Ry)
P(:,:,1:end-1,:)=P(:,:,1:end-1,:)+(V(:,:,1:end-1,:)-V(:,:,1:end-2,:)).^2./(2.*Ry);
P(:,:,2:end,:)=P(:,:,2:end,:)+\(V(:,:,2:end,:)-V(:,:,1:end-1,:))\.^2./(2.*Ry);
% Power due to heat dissipation along z (Rz)
P(:,:,1:end-1,:)=P(:,:,1:end-1,:)+(V(:,:,1:end-1,:)-V(:,:,1:end-2,:)).^2./(2.*Rz);
P(:,:,2:end,:)=P(:,:,2:end,:)+\(V(:,:,2:end,:)-V(:,:,1:end-1,:))\.^2./(2.*Rz);
P=P./(Lx*Ly*Lz);
SOR (Successive Over Relaxation) Numerical Method → Solving Potential, Temp., in a convergence loop

At this point a matrix of 3-D maps has been created in terms of $V_{\text{ReRAM}}$, thermal conductance, and $\rho_0(\text{ReRAM})$

Finds a solution for V and T using 3D Fourier/ Poisson equations in the matrix form $Ax = b$:

$$A = \begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}, \quad x = \begin{bmatrix}x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix}, \quad b = \begin{bmatrix}b_1 \\
b_2 \\
\vdots \\
b_n
\end{bmatrix}$$

Splitting $A$ into: $A = D + L + U$

$$(D + \omega L)x = \omega b - [\omega U + (\omega - 1)D]x$$

Using a relaxation factor $\omega \rightarrow$ speeds up convergence of a slow-converging process

% Using Successive Overrelaxation Method finds an approximated solution in
% T for 3D Poisson/Fourier equations
% T is a matrix (Nx)x(Ny)x(Nz)
% Ax is a matrix (Nx-1)x(Ny)x(Nz)
% Ay is a matrix (Nx)x(Ny-1)x(Nz)
% Az is a matrix (Nx)x(Ny)x(Nz-1)
% P is a matrix (Nx)x(Ny)x(Nz-1)
% for each (i j k) bin, i=2:Nx-1, j=2:Ny-1, k=2:Nz-1:
% Ax(i-1,j,k)*T(i-1,j,k)-T(x(i-1,j,k))+Ax(i,j,k)*T(i,j,k)-T(x(i+1,j,k))=0
% Ay(i,j-1,k)*T(i,j-1,k)+Ay(i,j,k)*T(i,j,k)-T(x(i,j+1,k))=0
% Az(i,j,k-1)*T(i,j,k-1)+Az(i,j,k)*T(i,j,k)-T(x(i,j,k+1))=0
% P(i,j,k)*(x(i+1,j,k)-x(i-1,j,k))*(y(i,j+1,k)-y(i,j-1,k))*(z(i,j,k+1)-z(i,j,k-1))
% T(:,:,1) is fixed to T1, T(:,:,Nz) is fixed to T2
% T1 and T2 are 2 more unknowns (determined by the boundary
% conditions at Zmin,Zmax)
% A system of (Nx-2)x(Ny-2)x(Nz-2)+2 linear equations is obtained
% Boundary conditions:
% for Xmin,Xmax,Ymin,Ymax (transversal boundary conditions) the lateral
% heat exchange is neglected
% for Zmin,Zmax the transfer between Tbound1 and Tbound2 is described
% by Abound1 and Abound2
% In the case of temperature (Fourier stationary equation) $T(i,j,k)$ is the
% temperature, $P(i,j,k)$ the power density (W/m^3) dissipated in the ijk bin,
% $Ax,Ay,Az$ the
% inverse of the thermal resistances (mK/W) between two adjacent bins along
% respectively x,y,z
% In case of potential (Poisson equation) $T(i,j,k)$ is the
% potential, $P(i,j,k)$ the charge density (C/m^3) in the ijk bin, $Ax,Ay,Az$ the
% electrical capacitance (F) between two adjacent bins along
% respectively x,y,z
%
% INPUT MUST BE IN MESHGRID FORMAT !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

function [T]=SOR_3D(Lx,Ly,Lz,T0,Ax,Ay,Az,Abound1,Abound2,Tbound1,Tbound2,P)
    global numsims_par;
    eps=numsims_par.varMax_SOR;

    % Convert from meshgrid to ndgrid format
    Ax = permute(Ax,[2 1 3]);
    Ay = permute(Ay,[2 1 3]);
    Az = permute(Az,[2 1 3]);
    P = permute(P,[2 1 3]);
    T0 = permute(T0,[2 1 3]);
    w=1.77;

    XTRANSFER =1;
    YTRANSFER =1;
    ZTRANSFER =1;

    [Nx,Ny,Nz]=size(T0);

    % precalculation of Adiag (the diagonal part of the coefficient matrix A)
    % and of b (the vector of constant terms)
    Adiag=zeros(Nx,Ny,Nz);
    if ZTRANSFER ==1
        Adiag1=Abound1; Adiag2=Abound2;
    else
        Adiag1=0; Adiag2=0;
    end

    b=zeros(Nx,Ny,Nz);
    b1=Abound1*Tbound1; b2=Abound2*Tbound2;

    b1=b1+sum(reshape(P(2:Nx-1,2:Ny-1,1),1,[]))*(Lx*Ly*Lz);
    b2=b2+sum(reshape(P(2:Nx-1,2:Ny-1,Nz),1,[]))*(Lx*Ly*Lz);

    Adiag1=Adiag1+sum(reshape(Az(2:Nx-1,2:Ny-1,1),1,[]));
    Adiag2=Adiag2+sum(reshape(Az(2:Nx-1,2:Ny-1,Nz-1),1,[]));

    if XTRANSFER ==1
        Adiag(3:Nx,1:Ny,2:Nz-1)=Adiag(3:Nx,1:Ny,2:Nz-1)+Ax(2:Nx-1,1:Ny,2:Nz-1);
        Adiag(1:Nx-2,1:Ny,2:Nz-1)=Adiag(1:Nx-2,1:Ny,2:Nz-1)+Ax(1:Nx-2,1:Ny,2:Nz-1);
    end
    if YTRANSFER ==1
        Adiag(1:Nx,3:Ny,2:Nz-1)=Adiag(1:Nx,3:Ny,2:Nz-1)+Ay(1:Nx,2:Ny-1,2:Nz-1);
        Adiag(1:Nx,1:Ny-2,2:Nz-1)=Adiag(1:Nx,1:Ny-2,2:Nz-1)+Ay(1:Nx,1:Ny-2,2:Nz-1);
    end
    if ZTRANSFER ==1
        % Additional code here if needed
    end
\[
\text{Adiag}(2:Nx-1,2:Ny-1,2:Nz-1) = \text{Adiag}(2:Nx-1,2:Ny-1,2:Nz-1) + \text{Az}(2:Nx-1,2:Ny-1,2:Nz-1);
\]
\[
\text{Adiag}(2:Nx-1,2:Ny-1,2:Nz-1) = \text{Adiag}(2:Nx-1,2:Ny-1,2:Nz-1) + \text{Az}(2:Nx-1,2:Ny-1,2:Nz-1);
\]
\]
\]
\[
\text{b}(2:Nx-1,2:Ny-1,2:Nz-1) = \text{p}(2:Nx-1,2:Ny-1,2:Nz-1) * (Lx*Ly*Lz);
\]
\]
\]
\[
MEX = 1;
\]
\[
\text{if } MEX == 1
\]
\[
\% Mex function used: much faster
\]
\[
\text{T} = \text{SOR}_\text{Loop}(T0,Nx,Ny,Nz,\text{Ax},\text{Ay},\text{Az},\text{Adiag},\text{Adiag}1,\text{Adiag}2,\text{b},\text{b}1,\text{b}2,\text{XTRANSFER},\text{YTRANSFER},\text{ZTRANSFER},\text{numsims_par.n_max_cycles}_\text{T},\text{eps},\text{w});
\]
\[
\text{else}
\]
\[
\% Matlab subroutine used
\]
\[
\text{T} = \text{SOR}_\text{loop}_\text{sub}(T0,Nx,Ny,Nz,\text{Ax},\text{Ay},\text{Az},\text{Adiag},\text{Adiag}1,\text{Adiag}2,\text{b},\text{b}1,\text{b}2,\text{XTRANSFER},\text{YTRANSFER},\text{ZTRANSFER},\text{numsims_par.n_max_cycles}_\text{T},\text{eps},\text{w});
\]
\[
\% Convert back to meshgrid format
\]
\[
\text{T} = \text{permute}(\text{T},[2 1 3]);
\]
\[
\text{function}
\]
\[
\text{T} = \text{SOR}_\text{loop}_\text{sub}(T0,Nx,Ny,Nz,\text{Ax},\text{Ay},\text{Az},\text{Adiag},\text{Adiag}1,\text{Adiag}2,\text{b},\text{b}1,\text{b}2,\text{XTRANSFER},\text{YTRANSFER},\text{ZTRANSFER},\text{nstepmax},\text{eps},\text{w})
\]
\[
\text{T} = T0;
\]
\[
\text{DISPLAY}=1;
\]
\[
\text{nsteps}=0;
\]
\[
\text{stop}=0;
\]
\[
\text{new}_\text{T} = \text{zeros}(\text{Nx},\text{Ny},\text{Nz});
\]
\[
\text{while } (\text{nsteps}<\text{nstepmax}) \&\& \text{stop==0}
\]
\[
\text{temp}=0;
\]
\[
\text{if } \text{ZTRANSFER} == 1
\]
\[
\text{for } j=2:Ny-1
\]
\[
\text{for } i=2:Nx-1
\]
\[
\text{temp}=\text{temp}+\text{T}(i,j,2)*\text{Az}(i,j,1);
\]
\[
\text{end}
\]
\[
\text{end}
\]
\[
\text{end}
\]
\[
\text{new}_\text{T}(2:Nx-1,2:Ny-1,1)=(1-w)*\text{T}(2:Nx-1,2:Ny-1,1)+w* (\text{b}1+\text{temp})/\text{Adiag}1;
\]
\[
\% i e j dovrebbero andare da 2 a Nx-1 e da 2 a Ny-1 rispettivamente
\]
\[
\% ma per qualche motivo cosi' e' molto piu' veloce
\]
\[
\text{for } k=2:Nz-1
\]
\[
\text{for } j=1:Ny
\]
\[
\text{for } i=1:Nx
\]
\[
\text{temp}=0;
\]
\[
\text{if } \text{XTRANSFER} == 1
\]
\[
\text{if } i>2
\]
\[
\text{temp}=\text{temp}+\text{new}_\text{T}(i-1,j,k)*\text{Ax}(i-1,j,k);
\]
\[
\text{end}
\]
\[
\text{end}
\]
\[
\text{end}
\]
\[
\text{if } i<Nx-1
\]
temp = temp + T(i+1, j, k) * Ax(i, j, k);
end
end

if YTRANSFER == 1
if j > 2
    temp = temp + new_T(i, j, l-1, k) * Ay(i, j, l-1, k);
end
if j < Ny-1
    temp = temp + T(i, j+1, k) * Ay(i, j, k);
end
end

if ZTRANSFER == 1
if i > 1 && i < Nx && j > 1 && j < Ny
    temp = temp + new_T(i, j, k-1) * Az(i, j, k-1);
    temp = temp + T(i, j, k+1) * Az(i, j, k);
end
end
new_T(i, j, k) = (1-w)*T(i, j, k) + w*(b(i, j, k) + temp) / Adiag(i, j, k);
end
end
end
temp = 0;
if ZTRANSFER == 1
    for j = 2:Ny-1
        for i = 2:Nx-1
            temp = temp + new_T(i, j, Nz-1) * Az(i, j, Nz-1);
        end
    end
end
new_T(:, :, Nz) = (1-w)*T(:, :, Nz) + w*(b2 + temp) / Adiag2;

% The temperature does not change in proximity of the lateral boundaries (heat transfer is neglected)
new_T(1, :, :) = new_T(2, :, :);
new_T(Nx, :, :) = new_T(Nx-l, :, :);
new_T(:, 1, :) = new_T(:, 2, :);
new_T(:, Ny, :) = new_T(:, Ny-l, :);
nsteps = nsteps + 1;
var = abs(T-new_T);
umi = sum((var(:) < eps));

if numi == Nx*Ny*Nz || nsteps == nstepmax
    stop = 1;
    if DISPLAY == 1
        fprintf('Convergence reached by SOR algorithm, nsteps=%d\n', nsteps);
    end
    if nsteps == nstepmax
        warning('Convergence not reached in SOR algorithm');
    end
end

T = new_T;
end
// SOR Loop
/* mex Function SOR_Loop.c */
/////////////////////////////////////////////////////////////////////////
// mex function SOR_Loop.c
// Calling:
//
T=SOR_loop(T0,Nx,Ny,Nz,Ax,Ay,Az,Adiag,Adiag1,Adiag2,b,b1,b2,XTRANSFER,YTRANSFER
,ZTRANSFER,nstepmax,eps,w);
// Compile:
// mex -O -v SOR_Loop.c
/////////////////////////////////////////////////////////////////////////
#include "mex.h"
#include "matrix.h"
void SOR_loop(double * T,double * T0,int Nx,int Ny,int Nz,double * Ax,double *
Ay,double * Az,double * Adiag,double Adiag1,double Adiag2,double * b,double
b1,double b2,int XTRANSFER,int YTRANSFER,int ZTRANSFER,int nstepmax,double
eps,double w)
{
// Define macros to simplify 3D array indexing
#define T(i,j,k) T[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
#define T0(i,j,k) T0[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
#define Ax(i,j,k) Ax[(i-1)+((j-1)+(k-1)*Ny)*(Nx-1) ]
#define Ay(i,j,k) Ay[(i-1)+((j-1)+(k-1)*(Ny-1))*Nx ]
#define Az(i,j,k) Az[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
#define Adiag(i,j,k) Adiag[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
#define b(i,j,k) b[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
#define new_T(i,j,k) new_T[(i-1)+((j-1)+(k-1)*Ny)*Nx ]

double * new_T = (double *) malloc(Nx*Ny*Nz*sizeof(double));

int i=0;
int j=0;
int k=0;
int DISPLAY=1;
int nsteps=0;
int stop=0;
int numi=0;
double temp=0;

//printf("Nx=%d,Ny=%d,Nz=%d,
Adiag1=%e,Adiag2=%e,b1=%e,b2=%e,w=%e",Nx,Ny,Nz,Adiag1,Adiag2,b1,b2,w);
//T=T0;
for(k=1;k<=Nz;k++)
{
  for(j=1;j<=Ny;j++)
  {
    for(i=1;i<=Nx;i++)
    {
      // SOR Loop
      /* mex Function SOR_Loop.c */
      ///////////////////////////////////////////////////////////////////////////
      // mex function SOR_Loop.c
      // Calling:
      //
      T=SOR_loop(T0,Nx,Ny,Nz,Ax,Ay,Az,Adiag,Adiag1,Adiag2,b,b1,b2,XTRANSFER,YTRANSFER
      ,ZTRANSFER,nstepmax,eps,w);
      // Compile:
      // mex -O -v SOR_Loop.c
      ///////////////////////////////////////////////////////////////////////////

      #include "mex.h"
      #include "matrix.h"

      void SOR_loop(double * T,double * T0,int Nx,int Ny,int Nz,double * Ax,double *
      Ay,double * Az,double * Adiag,double Adiag1,double Adiag2,double * b,double
      b1,double b2,int XTRANSFER,int YTRANSFER,int ZTRANSFER,int nstepmax,double
      eps,double w)
      {
      // Define macros to simplify 3D array indexing
      
      // Define macros to simplify 3D array indexing
      
      #define T(i,j,k) T[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
      #define T0(i,j,k) T0[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
      #define Ax(i,j,k) Ax[(i-1)+((j-1)+(k-1)*Ny)*(Nx-1) ]
      #define Ay(i,j,k) Ay[(i-1)+((j-1)+(k-1)*(Ny-1))*Nx ]
      #define Az(i,j,k) Az[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
      #define Adiag(i,j,k) Adiag[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
      #define b(i,j,k) b[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
      #define new_T(i,j,k) new_T[(i-1)+((j-1)+(k-1)*Ny)*Nx ]

      #define T(i,j,k) T[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
      #define T0(i,j,k) T0[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
      #define Ax(i,j,k) Ax[(i-1)+((j-1)+(k-1)*Ny)*(Nx-1) ]
      #define Ay(i,j,k) Ay[(i-1)+((j-1)+(k-1)*(Ny-1))*Nx ]
      #define Az(i,j,k) Az[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
      #define Adiag(i,j,k) Adiag[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
      #define b(i,j,k) b[(i-1)+((j-1)+(k-1)*Ny)*Nx ]
      #define new_T(i,j,k) new_T[(i-1)+((j-1)+(k-1)*Ny)*Nx ]

      double * new_T = (double *) malloc(Nx*Ny*Nz*sizeof(double));
T(i,j,k)=T0(i,j,k);
}
}

//new_T=zeros(Nx,Ny,Nz);
for(k=1;k<=Nz;k++)
{
    for(j=1;j<=Ny;j++)
    {
        for(i=1;i<=Nx;i++)
        {
            new_T(i,j,k)=0;
        }
    }
}

while ((nsteps<nstepmax) && stop==0)
{
    temp=0;
    if(ZTRANSFER ==1)
    {
        for(j=2;j<=Ny-1;j++)
        {
            for(i=2;i<=Nx-1;i++)
            {
                temp=temp+T(i,j,2)*Az(i,j,1);
            }
        }
        //new_T(2:Nx-1,2:Ny-1,1)=(1-w)*T(2:Nx-1,2:Ny-1,1)+w*(b1+temp)/Adiag1;
        for(i=2;i<=Nx-1;i++)
        {
            for(j=2;j<=Ny-1;j++)
            {
                new_T(i,j,1)=(1-w)*T(i,j,1)+w*(b1+temp)/Adiag1;
            }
        }
    }

    // Main loop
    for(k=2;k<=Nz-1;k++)
    {
        for(j=1;j<=Ny;j++)
        {
            for(i=1;i<=Nx;i++)
            {
                temp=0;
                if(XTRANSFER ==1)
                {
                    if(i>2)
                    {
                        temp=temp+new_T(i-1,j,k)*Ax(i-1,j,k);
                    }
                    if(i<Nx-1)
                    {
                        temp=temp+T(i+1,j,k)*Ax(i,j,k);
                    }
                }
                if(YTRANSFER ==1)
                {
                    if(j>2)
                    {
                        temp=temp+new_T(i,j-1,k)*Ay(i,j-1,k);
                    }
                    if(j<Ny-1)
                    {
                        temp=temp+T(i,j+1,k)*Ay(i,j,k);
                    }
                }
                if(YTRANSFER ==0)
                {
                    temp=temp+T(i,j,k)+Az(i,j,k);
                }
            }
        }
    }
}
temp = temp + new_T(i, j-1, k) * Ay(i, j-1, k);
}
if (j < Ny-1)
{
    temp = temp + T(i, j+1, k) * Ay(i, j, k);
}
if (ZTRANSFER == 1)
{
    if (i > 1 && i < Nx && j > 1 && j < Ny)
    {
        temp = temp + new_T(i, j, k-1) * Az(i, j, k-1);
        temp = temp + T(i, j, k+1) * Az(i, j, k);
        // printf("Az1=%e, T1=%e, Az2=%e, T2=%e ", Az(i, j, k-1), new_T(i, j, k-1), Az(i, j, k), T(i, j, k+1));
    }
}
new_T(i, j, k) = (1-w) * T(i, j, k) + w * (b(i, j, k) + temp) / Adiag(i, j, k);
// printf("w=%e, b1=%e, temp=%e, Adiag1=%e ", w, b(i, j, k), temp, Adiag(i, j, k));
// printf("new_T=%e, T=%e, i=%d, j=%d, k=%d\n", new_T(i, j, k), T(i, j, k), i, j, k);
}
}
// new_T(:,:,Nz-1)=new_T(:,:,Nz-2)+new_T(:,:,Nz-2)*new_T(:,:,Nz-2)+new_T(:,:,Nz-1)*new_T(:,:,Nz-1);
// The temperature does not change in proximity of the lateral boundaries (heat transfer is neglected)
//new_T(1,:,:)=new_T(2,:,:);
//new_T(Nx,:,:)=new_T(Nx-1,:,:);
for (k=1; k<= Nz; k++)
{
    for (j=1; j<= Ny; j++)
    {
        new_T(1, j, k) = new_T(2, j, k);
    }
}
new_T(Nx,j,k)=new_T(Nx-1,j,k);
}
}
//new_T(:,1,:)=new_T(:,2,:);
//new_T(:,Ny,:)=new_T(:,Ny-1,:);
for(k=1;k<=Nz;k++)
{
    for(i=1;i<=Nx;i++)
    {
        new_T(i,1,k)=new_T(i,2,k);
        new_T(i,Ny,k)=new_T(i,Ny-1,k);
    }
}
nsteps++;

//var=abs(T-new_T);
//numi=0;
numi=0;
for(k=1;k<=Nz;k++)
{
    for(j=1;j<=Ny;j++)
    {
        for(i=1;i<=Nx;i++)
        {
            if(T(i,j,k)>new_T(i,j,k)-eps && T(i,j,k)<new_T(i,j,k)+eps )
            numi++;
        }
    }
}

if(numi==Nx*Ny*Nz||nsteps==nstepmax )
{
    stop=1;
    if(DISPLAY==1)
    printf("Convergence reached by SOR algorithm,
    nsteps=%d\n",nsteps);
    if(nsteps==nstepmax)
    printf("Convergence not reached in SOR algorithm, number bins not
    converged=%d\n",numi-Nx*Ny*Nz);
}

//T=new_T;
for(k=1;k<=Nz;k++)
{
    for(j=1;j<=Ny;j++)
    {
        for(i=1;i<=Nx;i++)
        {
            T(i,j,k)=new_T(i,j,k);
        }
    }
}

//printf("nsteps= %d,numi=%d\n",nsteps,numi)  ;
}

// Gateway routine
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
    double * T;
    double * T0, * Ax,* Ay,* Az,* b,* Adiag;

    double Adiag1, Adiag2, b1, b2, eps, w;
    int Nx, Ny, Nz, XTRANSFER, YTRANSFER, ZTRANSFER, nstepmax;

    mwSize dims[2];
    /* Assign pointers to each input and output. */

    T0 = mxGetData(prhs[0]);
    Nx = mxGetScalar(prhs[1]);
    Ny = mxGetScalar(prhs[2]);
    Nz = mxGetScalar(prhs[3]);
    Ax = mxGetData(prhs[4]);
    Ay = mxGetData(prhs[5]);
    Az = mxGetData(prhs[6]);
    Adiag = mxGetData(prhs[7]);
    Adiag1 = mxGetScalar(prhs[8]);
    Adiag2 = mxGetScalar(prhs[9]);
    b = mxGetData(prhs[10]);
    b1 = mxGetScalar(prhs[11]);
    b2 = mxGetScalar(prhs[12]);
    XTRANSFER = mxGetScalar(prhs[13]);
    YTRANSFER = mxGetScalar(prhs[14]);
    ZTRANSFER = mxGetScalar(prhs[15]);
    nstepmax = mxGetScalar(prhs[16]);
    eps = mxGetScalar(prhs[17]);
    w = mxGetScalar(prhs[18]);

    // dims=mxGetDimensions(T0);
    dims[0]=Nx;
    dims[1]=Ny;
    dims[2]=Nz;
    plhs[0]= mxCreateNumericArray(3, dims, mxDOUBLE_CLASS, mxREAL);
    /* create a C pointer to a copy of the output matrix */
    T = mxGetPr(plhs[0]);
    /* call the C subroutine */

    SOR_loop(T, T0, Nx, Ny, Nz, Ax, Ay, Az, Adiag, Adiag1, Adiag2, b, b1, b2, XTRANSFER, YTRANSFER,
             ZTRANSFER, nstepmax, eps, w);
}

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

function [time out vs t transitions trec rec_pos nrec tSpacer Spacer_pos nSpacer ngen lifetime VacMap2 In sexflag

    r_GB]=Ions_DD_kMC(TurnOFFGEN_Recomb_STOP, VpinChargeFlag, SetFlag, FormingFlag, RestFlag, sexFlag, Rcutoff, VFlag, PULSE_OFF, IcomplMeet, restriction, FlagAll,

    ions_map, VacMap0, VacMap2, Edev, Tdev, max_time, nSTEPS, Btrec, Brec_pos, Bnrec,

    BtSpacer, BSpacer_pos, BnSpacer)

    global Xd Yd Zd;
    global X1_D X2_D Y1_D Y2_D layers;
nsteps_max=nSTEPS;
ndiff_max=100000;
flagAVG=0;

C=.4*lambda;%rate of increasing GB size
Default_r_GB=5e-10;
.radialMax0,iradialmax,iradialmin,radialMin2,r_GB, VacMap0_new, VacMap2_new, Out, In, OEL]=modVacMaps(VacMap0,VacMap2,ions_map,Default_r_GB,C,restriction);
VacMap0=VacMap0_new;
VacMap2=VacMap2_new;
fprintf('
 r_GB=%d, 
',r_GB);
fprintf('
 r_v_max=%d, r_i_max(MeOx)=%d;   r_i_min(MeOx)=%d
 r_Pin_min=%d
',radialMax0,iradialmax,iradialmin,radialMin2);
fprintf('
 %%Ions in Bulk=%d; CF=%d; OEL=%d 
',Out,In,OEL);

[Xmesh Ymesh Zmesh] = meshgrid(Xd,Yd,Zd);
Fx=Edev.Exmesh;
Fy=Edev.Eymesh;
Fz=Edev.Ezmesh;
Fmesh=sqrt(Fx.^2+Fy.^2+Fz.^2);
Edev.Fmesh=Fmesh;

% Efeild and Temp used for Defect generation rates
Fgen=interp3_linear(Edev.Xmesh,Edev.Ymesh,Edev.Zmesh,Fmesh,Xmesh,Ymesh,Zmesh ,NaN);
Tgen=interp3_linear(Tdev.Xmesh,Tdev.Ymesh,Tdev.Zmesh,Tdev.Tmesh,Xmesh,Ymesh,Zmesh,Z mesh,NaN);
%-------------------------Defining Generation Ea values-------------------%
%GB region
if ResetFlag==1
 fprintf('
 Reset Fast DC Sweep conditions
');
 [vGB kGB]=b_genBULK;%modifed
 Ea_effective=Eact_genBULK-vGB.*Fgen;%modifed
 Ea_effective(Ea_effective<0)=0;
 G=G0_gen.*exp(-q.*(Ea_effective)./(Kb.*Tgen));
 %Bulk region
 [vBULK kBULK]=b_genBULK;
 Ea_effective2=Eact_genBULK-vBULK.*Fgen;
 Ea_effective2(Ea_effective2<0)=0;
 G(Xmesh.^2+Ymesh.^2>r_GB.^2)=G0_gen.*exp(-
 q.*(Ea_effective2(Xmesh.^2+Ymesh.^2>r_GB.^2))./(Kb.*Tgen(Xmesh.^2+Ymesh.^2>
 r_GB.^2)));

 if FlagAll==1
 GEN=0;  % turns on Generation
 REC=1;  % recomb
 DIFF=1;  % diff
 end

 STOP_ON_GEN=1;  %enables prgroam to stop of genreation event
 STOP_ON_REC=1;
 DEF_OVERLAP=0;  %disabled overlap of vac when =1,
 Ngen_stop=3;  %determine number of generation event to allow before
 reclamation of SOR
 Nrec_stop=3;
 nrecSTOP=0;% keep this at 0 it initalze stop count

 %ions_mapOUT=ions_map;
 [ ions_map ]=TurnOFF_Selected_IonDiff(ions_map);
end

%GB region
if FormingFlag==1 || SetFlag==1
  if FormingFlag==1; fprintf(' Forming CVF conditions
'); end;
  if SetFlag==1; fprintf(' Set Fast DC Sweep conditions, gen=5
'); end;
  [vGB kGB]=b_genGB;
  Ea_effective=Eact_genGB-vGB.*Fgen;
  Ea_effective(Ea_effective<0)=0;
  G=G0_gen.*exp(-q.*(Ea_effective)/(Kb.*Tgen));
  % Bulk region
  [vBULK kBULK]=b_genBULK;
  Ea_effective2=Eact_genBULK-vBULK.*Fgen;
  Ea_effective2(Ea_effective2<0)=0;
  G(Xdmesh.^2+Ydmesh.^2>r_GB.^2)=G0_gen.*exp(-
    q.*(Ea_effective2(Xdmesh.^2+Ydmesh.^2>r_GB.^2))/
    (Kb.*Tgen(Xdmesh.^2+Ydmesh.^2>r_GB.^2)));
  if FlagAll==1
    GEN=1;  % turns on Generation
    REC=1;  % recomb
    DIFF=1; % diff
  end
  STOP_ON_GEN=1; %enables prgroom to stop of generation event
  STOP_ON_REC=1;
  DEF_OVERLAP=0; %disabled overlap of vac when =1,
  if SetFlag==1
    Ngen_stop=5;
    Nrec_stop=5;
  else %determine number of generation event to allow before reclamation of SOR
    Ngen_stop=5;
    Nrec_stop=5;
  end
  nrecSTOP=0;
  end
  if GEN==1
    fprintf(' Generation enabled
');
  else
    fprintf(' Generation disabled
');
  end
  if REC==1
    fprintf(' Reccomb enabled
');
  else
    fprintf(' Recomb disabled
');
  end
%Cluster of Hf
%finds dist between vac and nearest neighbor
% distx= bsxfun(@minus,vac_map(:,1),Xdmesh(:)');
% disty= bsxfun(@minus,vac_map(:,2),Ydmesh(:)');
% distz= bsxfun(@minus,vac_map(:,3),Zdmesh(:)');
% dist=sqrt(distx.^2+disty.^2+distz.^2);
% %makes G = G_cluster for every sphere centered on a vac position with 20 or
% more other vac within .5nm radius
% B=dist<.5e-9;
% V=sum(B,1) >=32;
% Ea_effective3=Eact_genGB-b_genGB.*Fgen;
% Ea_effective3(Ea_effective3<0)=0;
% G(V)=G0_gen.*exp(-q.*(Ea_effective3(V))/(Kb.*Tgen(V)));
%-------------------------------------------------------------------------%
%------------------------Defining Recombination Ea Values-----------------%
% Recombination rates
R=G0_gen.*exp(-q.*Eform)./(Kb.*Tgen);
R(Xdmesh.^2+Ydmesh.^2>r_GB.^2)=G0_gen.*exp(-q.*Eform2)./(Kb.*Tgen(Xdmesh.^2+Ydmesh.^2>r_GB.^2));

[Nions,~]=size(ions_map);
[Nvac0,~]=size(VacMap0);
[Nvac2,~]=size(VacMap2);

%-------------------------------------------------------------------------%
if (DEF_OVERLAP == 0)
distmax=lambda;
Xdmesh_=reshape(Xdmesh,[1,size(Xdmesh)]);
Ydmesh_=reshape(Ydmesh,[1,size(Ydmesh)]);
Zdmesh_=reshape(Zdmesh,[1,size(Zdmesh)]);

% Forbid to generate vacancies at a distance \leq distmax from % other vacancies
% this takes up a lot of memory
if (Nions+Nvac0>0)
  Xdef=reshape([vac_map(:,1);ions_map(:,1)],[],1,1,1);
  Ydef=reshape([vac_map(:,2);ions_map(:,2)],[],1,1,1);
  Zdef=reshape([vac_map(:,3);ions_map(:,3)],[],1,1,1);
  Xdef=reshape(VacMap0(:,1),[],1,1,1);
  Ydef=reshape(VacMap0(:,2),[],1,1,1);
  Zdef=reshape(VacMap0(:,3),[],1,1,1);

dist=bsxfun(@minus,Xdef,Xdmesh_).^2+bsxfun(@minus,Ydef,Ydmesh_).^2+bsxfun(@minus,Zdef,Zdmesh_).^2;
V=any(dist<distmax^2,1);
G(V)=0;
end
%--------------------Initalizing Loop-------------------------------------%
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%

angle_xy=[0 22.5 45 67.5 90 135 157.5 180 202.5 225 247.5 270 292.5 315 337.5]*pi/180; % if increase the number of angles , same as allowing interstial movement
angle_z= [0 22.5 45 67.5 90 135 157.5 180]*pi/180;
% angle_xy=[0 45 90 135 180 225 270 315]*pi/180; % if increase the number of angles , same as allowing interstial movement
% angle_z= [0 90 180]*pi/180;
tot_time=0;
time=zeros(1,nsteps_max+1);
transitions=zeros(nsteps_max,3);
time(1)=0;
out_vs_t(1).Ions=ions_map;
out_vs_t(1).Vac=VacMap0;

% keyboard;
n=1;
nrec=Bnrec;         rec_pos=Brec_pos;       trec=Btrec;
nSpacer=BnSpacer;   Spacer_pos=BSpacer_pos; tSpacer=BTSpacer;
nreg=0;
stop=0;
ntransitions=0;
ndiff=0;
% generate array of random numbers to be used in MonteCarlo algorithm
r=random('unif',0,1,nsteps_max,2);
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
while  stop==0
  % Diffusion rates
  if DIFF==1 && ndiff<ndiff_max
    if flagAVG~=1
      [Diff_rates,~,~,~]=Ions_Rates_DD(VpinChargeFlag,Tdev,ions_map,VacMap0,VacMap2,E
devel,angle_xy,angle_z,IcomplMeet,kBULK,kGB,r_GB);
% end
    else
      Diff_rates=[];
% end
  end
  Diff_rates(isnan(Diff_rates))=0;
  % Generation rates
  if GEN==1
    Gen_rates=G;
  else
    Gen_rates=[];
  end
  % keyboard;
  % Recombination
  if REC==1 && Nions>0 && Nvac0>0
    AllVacMap=[VacMap0; VacMap2];
    distx= bsxfun(@minus,ions_map(:,1),AllVacMap(:,1)');% Matrix with
distances between traps and ions
    disty= bsxfun(@minus,ions_map(:,2),AllVacMap(:,2)');
    distz= bsxfun(@minus,ions_map(:,3),AllVacMap(:,3)');
    dist=sqrt(distx.^2+disty.^2+distz.^2);
    Rec=interp3_linear(Xdmesh,Ydmesh,Zdmesh,R,AllVacMap(:,1),AllVacMap(:,2),AllVacM
    ap(:,3),NaN); %grabes relavent R values
    if FormingFlag==1 || SetFlag==1
      Rec_rates=bsxfun(@times,Rec',heaviside(1*lambda-dist)); % heaviside(x)
      has the value 0 for x < 0, 1 for x > 0, and 0.5 for x = 0.
      elseif ResetFlag==1
        m = ...
        (1) .* (dist <= 2.5e-10 ) + ...
        (exp(-dist./(.5*lambda)+2)) .* (dist > 2.5e-10 ) ;
        Rec_rates=bsxfun(@times,Rec',m);
% end
      Rec_rates=bsxfun(@times,Rec',heaviside(1*lambda-dist)); % heaviside(x)
      has the value 0 for x < 0, 1 for x > 0, and 0.5 for x = 0.
% The difference is that heaviside--> the recombination is possible only
% when the ion and the vacancy are at a distance < lambda
% Rec_rates(exp) considers the recombination rate decreasing
% exponentially with the distance between the ion and the vacancy.
% fprintf('
 Recomb max Rate=%d\n',max(Rec_rates(:)));
    else
      Rec_rates=[];
% end
  end

X=1000;
if ((n==X || flagAVG==1) && VFlag==0)
  flagAVG=1;
  %Diff_ratesOrginal=Diff_rates;
  CuttOff_loc = Diff_rates > Rcutoff;
  Diff_rates(CuttOff_loc) = 0;
% end
\[ R_{kMC} = \{ \text{Diff} \_\text{rates}(:, :)', \ \text{Gen} \_\text{rates}(:, :)', \ \text{Rec} \_\text{rates}(:, :)' \}; \% \text{Combines all Rates into one array} \]

\[ R_{kMC}(\text{isnan}(R_{kMC})) = 0; \% \text{this is a fix for adding spacer... should do another approach for long term} \]

\[
\text{if } \text{numel}(R_{kMC}) \geq 0 \\
\{ \text{nextevent, } \text{lifetime} \} = \text{kMC\_nextevent}(R_{kMC}, r(n,:)); \% \text{k\_MC nextevent} \\
\quad \% \text{keyboard}; \\
\quad \% \text{if } \text{tot\_time}+\text{lifetime} \geq \text{max\_time} \\
\quad \% \text{keyboard};
\]

\[
\text{if nextevent} \leq \text{numel(Diff\_rates)}
\]

\[ \% \text{Ion Drift/Diffusion} \]

\[ [j \ i \ k] = \text{ind2sub(size(Diff\_rates),nextevent)}; \]

\[ \text{ndiff} = \text{ndiff}+1; \% \text{keyboard}; \]

\[
\text{if } i \leq \text{Nions} \\
\quad \% \text{Ion diffusion} \\
\text{transitions(n,:) = [i \ \text{angle\_xy(j)}*180/pi \ \text{angle\_z(k)}*180/pi]};
\]

\[ \text{ions\_map(i,1)} = \text{ions\_map(i,1)} + \lambda \cdot \cos(\text{angle\_xy(j)}) \cdot \sin(\text{angle\_z(k)}); \]

\[ \text{ions\_map(i,2)} = \text{ions\_map(i,2)} + \lambda \cdot \sin(\text{angle\_xy(j)}) \cdot \sin(\text{angle\_z(k)}); \]

\[
\text{ions\_map(i,3)} = \text{ions\_map(i,3)} + \lambda \cdot \cos(\text{angle\_z(k)}); \\
\text{if } (n==1 || n==3 || n==5 || n==10 || n==50 || n==60 || n==75 || n==90 || n==100 || n==150 || n==200 || n==400 || n==500 || n==750 || n==1000 || n==1500 || n==2000 || n==2500 || n==3000 || n==4000 || n==4500 || n==5000 || n==6000 || n==7000 || n==8000 || n==9000 || n==10000 || n==15000 || n==20000 || n==4500 || n==5000 || n==7500 || n==9500) \\
\text{fprintf('t=%d: O Ion Diffusion,n=%d; pos=[%d,%d,%d] rate=%d, Rcut=%d
',tot\_time,n,ions\_map(i,1),ions\_map(i,2),ions\_map(i,3),R_{kMC}(\text{nextevent }),R_{cutoff});} \% \text{keyboard};
\]

\[
\text{if } \text{R\_kMC(\text{nextevent})} < 2e12
\quad \text{fprintf('t=%d: O Ion Diffusion,n=%d; pos=[%d,%d,%d] rate=%d\n',tot\_time,n,ions\_map(i,1),ions\_map(i,2),ions\_map(i,3),R_{kMC}(\text{nextevent }));}
\]

\[
\% \text{end} \\
\% \text{----for testing-----------------------------} \\
\% \text{ions\_map(i,1)} = -1.0e-8-2.5e-10; \\
\% \text{---------------------------------------------} \\
\text{flag}=0; \\
\text{if } \text{ions\_map(ions\_map(:,1)} < \text{X1\_D)} \\
\text{fprintf('t=%d: This Ion Diffused into Spacer Region,n=%d\n',tot\_time,n);} \\
\quad \text{flag}=1; \\
\text{end}
\]

\[
\text{if } \text{ions\_map(ions\_map(:,1)} > \text{X2\_D) \\
\text{fprintf('t=%d: This Ion Diffused into Spacer Region,n=%d\n',tot\_time,n);} \\
\]
flag=1;

end

if ions_map(ions_map(:,2)<Y1_D)
fprintf('t=%d: This Ion Diffused into Spacer Region,n=%d
',tot_time,n);
flag=1;
end

if ions_map(ions_map(:,2)>Y2_D)
fprintf('t=%d: This Ion Diffused into Spacer Region,n=%d
',tot_time,n);
flag=1;
end

if flag==1;
% Save the time and pos of where ion entered spacer
tSpacer(nSpacer)=tot_time;
Spacer_pos(nSpacer,:)=ions_map(i,:);
nSpacer=nSpacer+1;
% Remove the ion from Ion Map
ions_map=[ions_map(1:i-1,:);ions_map(i+1:end,:)];
end
else
% Vacancy diffusion
if i<= (Nvac0+Nions)
    transitions(n,:)=[i angle_xy(j)*180/pi angle_z(k)*180/pi];
    VacMap0(i-Nions,1)=VacMap0(i-Nions,1)+lambda.*cos(angle_xy(j)).*sin(angle_z(k));
    VacMap0(i-Nions,2)=VacMap0(i-Nions,2)+lambda.*sin(angle_xy(j)).*sin(angle_z(k));
    VacMap0(i-Nions,3)=VacMap0(i-Nions,3)+lambda.*cos(angle_z(k));
    fprintf('t=%d: Vacancy Diffusion,n=%d; pos=[%d,%d,%d]
',tot_time,n,VacMap0(i-Nions,1),VacMap0(i-Nions,2),VacMap0(i-Nions,3));
end
if (i> Nvac0+Nions) &&
  (i<= (Nions+Nvac0+length(VacMap2(:,1)))))
    transitions(n,:)=[i angle_xy(j)*180/pi angle_z(k)*180/pi];
    VacMap2(i-(Nions+Nvac0),1)=VacMap2(i-(Nions+Nvac0),1)+lambda.*cos(angle_xy(j)).*sin(angle_z(k));
    VacMap2(i-(Nions+Nvac0),2)=VacMap2(i-(Nions+Nvac0),2)+lambda.*sin(angle_xy(j)).*sin(angle_z(k));
    VacMap2(i-(Nions+Nvac0),3)=VacMap2(i-(Nions+Nvac0),3)+lambda.*cos(angle_z(k));
    fprintf('t=%d: Vacancy Diffusion2,n=%d; pos=[%d,%d,%d]
',tot_time,n,VacMap2(i-(Nions+Nvac0),1),VacMap2(i-(Nions+Nvac0),2),VacMap2(i-(Nions+Nvac0),3));
end
end
end

% Ion/Vacancy pair generation
if nextevent>numel(Diff_rates)&&
nextevent<=numel(Diff_rates)+numel(Gen_rates)

% J AND I MUST BE SWITCHED BECAUSE MESHGRID IS USED INSTEAD OF NDGRID
\[ j \text{ and } k = \text{ind2sub(size(Gen_rates),nextevent-numel(Diff_rates))}; \]

% Random selection of position inside the selected cell
\[ rx = \text{random('unif', conditional(i>1,-0.5,0),1,1)}; \]
\[ ry = \text{random('unif', conditional(j>1,-0.5,0),1,1)}; \]
\[ rz = \text{random('unif', conditional(k>1,-0.5,0),1,1)}; \]

\[ x_{\text{vac}} = Xd(i) + rx(1) * (Xd(2) - Xd(1)); \]
\[ y_{\text{vac}} = Yd(j) + ry(1) * (Yd(2) - Yd(1)); \]
\[ z_{\text{vac}} = Zd(k) + rz(1) * (Zd(2) - Zd(1)); \]

% Ion placed at a distance \(0.5\lambda\) from the vacancy:
% interstitial placement of ion for first jump
% Random polar coordinates
\[ r_{\text{pol}} = [2\pi \pi].* \text{rand}(1,2); \]
\[ x_{\text{ion}} = x_{\text{vac}} + 0.5 \lambda \cos(r_{\text{pol}}(1)) \sin(r_{\text{pol}}(2)); \]
\[ y_{\text{ion}} = y_{\text{vac}} + 0.5 \lambda \sin(r_{\text{pol}}(1)) \sin(r_{\text{pol}}(2)); \]
\[ z_{\text{ion}} = z_{\text{vac}} + 0.5 \lambda \cos(r_{\text{pol}}(1)); \]
% makes sure ion is generated inside the ReRAM
\[ x_{\text{ion}}(x_{\text{ion}}<X1_D)=X1_D; \]
\[ x_{\text{ion}}(x_{\text{ion}}>X2_D)=X2_D; \]
\[ y_{\text{ion}}(y_{\text{ion}}<Y1_D)=Y1_D; \]
\[ y_{\text{ion}}(y_{\text{ion}}>Y2_D)=Y2_D; \]
\[ z_{\text{ion}}(z_{\text{ion}}<0)=0; \]
\[ z_{\text{ion}}(z_{\text{ion}}>\text{sum([layers(:,t)])})=\text{sum([layers(:,t)])}; \]
% Add a new ion
\[ \text{ions\_map}(\text{Nions}+1,:)=[x_{\text{ion}} y_{\text{ion}} z_{\text{ion}}]; \]
% Add a new vacancy
\[ \text{VacMap0}(:,\text{Nvac0}+1,:)=[x_{\text{vac}} y_{\text{vac}} z_{\text{vac}}]; \]
\[ \text{Nions}=\text{Nions}+1; \]
\[ \text{Nvac0}=\text{Nvac0}+1; \]
\[ \text{ngen}=\text{ngen}+1; \]
\[ \text{fprintf}('t=%d: Generation,n=%d; pos=[%d,%d,%d], rate=%d
', \text{tot\_time}, \text{ngen}, \text{VacMap0}(\text{end},1), \text{VacMap0}(\text{end},2), \text{VacMap0}(\text{end},3), \text{R\_kMC(nextevent)}); \]
% Ion/Vacancy pair recombination
\[ \text{if} \ \text{STOP\_ON\_GEN} \ \&\& \ \text{ngen}==\text{Ngen\_stop} \]
\[ \ \text{stop}=1; \]
\[ \text{end} \]
% Ion/Vacancy pair recombination
\[ [i \ j] = \text{ind2sub(size(Rec\_rates),nextevent-numel(Diff\_rates)-numel(Gen\_rates))}; \]
% Save the time and pos of where the recombination happened
\[ \text{trec}(\text{nrec})=\text{tot\_time}; \]
\[ \text{rec\_pos}(\text{nrec},:)=\text{ions\_map}(i,:); \]
\[ \text{nrec}=\text{nrec}+1; \]
\[ \text{nrecSTOP}=\text{nrecSTOP}+1; \]
% keyboard;
\[ \text{fprintf}('t=%d: Recombination,n=%d; pos=[%d,%d,%d], rate=%d
', \text{tot\_time}, \text{ngen}, \text{ions\_map}(i,1), \text{ions\_map}(i,2), \text{ions\_map}(i,3), \text{R\_kMC(nextevent)}); \]
% Remove the ion
\[ \text{ions\_map}=[\text{ions\_map}(1:i-1,:);\text{ions\_map}(i+1:end,:)]; \]
% Remove the vacancy
\[ \text{if} \ \ (j-1) > \text{length(\text{VacMap0}(:,1))} \]
\[ \ \%keyboard; \]
\[ \text{AAAC}=(j)-\text{length(\text{VacMap0}(:,1))}; \]
\[ \text{VacMap2}=[\text{VacMap0}(\text{AAAC}-1,:);\text{VacMap0}(\text{AAAC}+1:end,:)]; \]
\[ \text{Nions}=\text{Nions}-1; \]

231
% keyboard;
else
    VacMap0=[VacMap0(1:j-1,:);VacMap0(j+1:end,:)];
    Nions=Nions-1; Nvac0=Nvac0-1;
end

if STOP_ON_REC && nrecSTOP==Nrec_stop
    stop=1;
end
else

    if PULSE_OFF==1 %Vramp is on
        tot_time=max_time;
    elseif PULSE_OFF==0 %Pulse Mode
        stop=1;
    end
else
    % keyboard;
    tot_time= max_time;
end

if n>=nsteps_max || tot_time>=max_time
    stop=1;
    if tot_time<max_time
        warning('Maximum simulation time not reached during drift/diffusion simulations');
    end
end
end

n=n+1;
time(n)=tot_time;
out_vs_t(n).Ions=ions_map;
out_vs_t(n).Vac=VacMap0;
if exist('trec','var')==1
    out_vs_t(n).RecombTime=trec;
    out_vs_t(n).RecombPos=rec_pos;
else
    out_vs_t(n).RecombTime=[];
    out_vs_t(n).RecombPos=[];
end
if exist('tSpacer','var')==1
    out_vs_t(n).SpacerIonsPos=Spacer_pos;
    out_vs_t(n).SpacerIonsTime=tSpacer;
else
    out_vs_t(n).SpacerIonsPos=[];
    out_vs_t(n).SpacerIonsTime=[];
end

if n==(nsteps_max)
    sexflag=1
end; %det if max sim time was reached, indicating 1 of the prerequisues
for Rcutoff to begin
if n<(nsteps_max)
    sexflag=0;
end;
end
time = time(1:n);
transitions = transitions(1:ntransitions,:);

end

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

function [radialMax0, iradialmax, iradialmin, radialMin2, r_GB, VacMap0_out, VacMap2_out Out In OEL] =
modVacMaps(VacMap0,VacMap2,ions_map,Default_r_GB,C,restriction)

% MODVACMAPS Summary of this function goes here
% Detailed explanation goes here
% keyboard;

if numel(VacMap2)~=0
 [~, rhoVmap0,~] = cart2pol(VacMap0(:,1),VacMap0(:,2),VacMap0(:,3));
 [~, rhoVmap2,~] = cart2pol(VacMap2(:,1),VacMap2(:,2),VacMap2(:,3));
 [~, rhoimap,zzionmap] = cart2pol(ions_map(:,1),ions_map(:,2),ions_map(:,3));
 L = length(rhoinmap);
 radialMax0 = max(rhoVmap0);
 radialMin2 = min(rhoVmap2);

 noOELionmap_rad = rhoimap(zzionmap<5e-9);
 iradialmin = min(noOELionmap_rad);
 iradialmax = max(noOELionmap_rad);

 FLaG = 0;

 if restriction==0
     if radialMax0>Default_r_GB
         r_GB = radialMax0+C;
         fprintf('\n AA \n');
         for inxt=1:length(rhoVmap2(:,1))
             if rhoVmap2(inxt)<r_GB %inside =0
                 % keyboard;
                 VacMap0=[VacMap0;VacMap2(inxt,:)];
                 VacMap2(inxt,:) = Inf;
                 FLaG=1;
             end
         end
     end
     if FLaG==1
         VacMap2(VacMap2==Inf)=[];
         VacMap2=reshape(VacMap2,length(VacMap2(1,:))/3,3);
         [~, rhoVmap0,~] = cart2pol(VacMap0(:,1),VacMap0(:,2),VacMap0(:,3));
         [~, rhoVmap2,~] = cart2pol(VacMap2(:,1),VacMap2(:,2),VacMap2(:,3));
         radialMax0 = max(rhoVmap0);
         radialMin2 = min(rhoVmap2);
         fprintf('\n A \n');
     end
     else
         r_GB = Default_r_GB; % default GB radius
         fprintf('\n B \n');
         for inxt=1:length(rhoVmap2(:,1))
             if rhoVmap2(inxt)<r_GB % inside =0
                 % keyboard;
                 VacMap0=[VacMap0;VacMap2(inxt,:)];
             end
         end
     end

else

end

end

233
VacMap2\( (\text{inxt},:) = \text{Inf}; \)
FLaG=1;
end
endif
if FLaG==1
VacMap2 (VacMap2==Inf)=[];
VacMap2=reshape(VacMap2,length(VacMap2(1,:))/3,3);
[~, rhoVmap0,~]=cart2pol(VacMap0(:,1),VacMap0(:,2),VacMap0(:,3));
[~, rhoVmap2,~]=cart2pol(VacMap2(:,1),VacMap2(:,2),VacMap2(:,3));
radialMax0=max(rhoVmap0);
radialMin2=min(rhoVmap2);
fprintf('\n C \n');
end
endif
elseif restriction==1
r_GB=Default_r_GB; %default GB radius
fprintf('\n D \n');
for inxt=1:length(rhoVmap2(:,1))
if rhoVmap2(inxt)<r_GB %inside =0
VacMap0=[VacMap0;VacMap2(inxt,:)];
VacMap2(inxt,:)=Inf;
FLaG=1;
end
endif
if FLaG==1
VacMap2 (VacMap2==Inf)=[];
VacMap2=reshape(VacMap2,length(VacMap2(1,:))/3,3);
[~, rhoVmap0,~]=cart2pol(VacMap0(:,1),VacMap0(:,2),VacMap0(:,3));
[~, rhoVmap2,~]=cart2pol(VacMap2(:,1),VacMap2(:,2),VacMap2(:,3));
radialMax0=max(rhoVmap0);
radialMin2=min(rhoVmap2);
fprintf('\n E \n');
end
end
end

%keyboard;
if restriction==0
ion_inCF=rhoimap(rhoimap<r_GB & zzionmap<5e-9);
ion_MeOx=rhoimap(rhoimap>=r_GB & zzionmap<5e-9);
ion_OEL=rhoimap(zzionmap>5e-9);
In=length(ion_inCF(:,1));
Out=length(ion_MeOx(:,1));
OEL=length(ion_OEL(:,1));
In=100*In/L;
Out=100*Out/L;
OEL=100*OEL/L;
%fprintf('\n %Ions in Bulk=%d; CF=%d; OEL=%d \n');
end;
if restriction==1
ion_inCF=NaN;
ion_MeOx=NaN;
ion_OEL=NaN;
In=NaN;
Out=NaN;
OEL=NaN;
%fprintf('\n %Ions in Bulk=%d; CF=%d; OEL=%d \n');
end;
end;

VacMap0_out=VacMap0;
VacMap2_out=VacMap2;

end

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

function [ ions_mapOUT ] = TurnOFF_Selected_IonDiff(ions_mapIN)

%TURNOFF_SELECTED_IONDIFF Summary of this function goes here
% Detailed explanation goes here

ions_map=ions_mapIN;

% [~, r, zz] = cart2pol(ions_map(:,1), ions_map(:,2), ions_map(:,3));
% keyboard;

for inxt=1:length(ions_map(:,1))
    if ions_map(inxt,3)>5e-9 % inside = 0
        ions_map(inxt,:)=Inf;
    end
end

% keyboard;
ions_map(ions_map==Inf)=[ ];

if numel(ions_map)==numel(ions_mapIN)
    ions_map=reshape(ions_map,length(ions_map(1,:))/3,3);
end

ions_mapOUT=ions_map;
end
function [DiffR,xxp1,yyp1,zzp1]=
Ions_Rates_DD(VpinChargeFlag,Tdev,ions_map,vac_map0,vac_map2,E_ext_dev,angle_xy,
angle_z, IcomplMeet,kBULK,kGB,r_GB)

global X1_D X2_D Y1_D Y2_D layers;
global Xd Yd Zd;

[Nions ~]=size(ions_map);
[Nvac0 ~]=size(vac_map0);
[Nvac2 ~]=size(vac_map2);

%-------------------------Convert to spherical coordinates-----------------
[xxp1,angle_xy_,angle_z_]=meshgrid([ions_map(:,1);vac_map0(:,1);vac_map2(:,1)],
angle_xy,angle_z);
[yyp1,~,~]=meshgrid([ions_map(:,2);vac_map0(:,2);vac_map2(:,2)],angle_xy,angle_z);
[zzp1,~,~]=meshgrid([ions_map(:,3);vac_map0(:,3);vac_map2(:,3)],angle_xy,angle_z);

%-------------------------Charge of each ion or vac------------------
[Q,~]=meshgrid([Q_i*ones(Nions,1);Q_v*ones(Nvac0,1);Q_v2*ones(Nvac2,1)],angle_xy,angle_z);

% Diffusion (Ion or Vac)

$$D = G_0 \cdot e^{-q^* \left( \frac{E_a - Q^* \frac{\lambda^2}{2} \left[ F_{int} + F_{ext} \right]}{K_b \cdot T} \right)}$$

Ionic Charge = Fint = $-\frac{qQ}{4 \pi \epsilon_0 k r}$

- $F_{int}$ Columbic Repulsion
- $F_{ext}$ External E-Field = $-\nabla$Potential

- Black circled ion's effect on surrounding neighbors in terms of vectors

\[ \rightarrow + \rightarrow \]
Ea_i=zeros(Nions,1);
Ea_i(ions_map(:,3)<=layers(1).t)=Ea_i_lay1;%MeOx
Ea_i(ions_map(:,3)>layers(1).t)=Ea_i_lay2; %OEL

%---------------------------Diffusion Ea_ions Spacer----------------------%
Ea_i(ions_map(:,1)<X1_D)=Ea_i_Spacer;
Ea_i(ions_map(:,1)>X2_D)=Ea_i_Spacer;
Ea_i(ions_map(:,2)<Y1_D)=Ea_i_Spacer;
Ea_i(ions_map(:,2)>Y2_D)=Ea_i_Spacer;

Ea_v=zeros(Nvac0+Nvac2,1);
Ea_v([vac_map0(:,3);vac_map2(:,3)])<=layers(1).t)=Ea_v_lay1;
Ea_v([vac_map0(:,3);vac_map2(:,3)])>layers(1).t)=Ea_v_lay2;

[Ea,~,~]=meshgrid([Ea_i;Ea_v],angle_xy,angle_z);
xxp2=xxp1+lambda.*cos(angle_xy_).*sin(angle_z_);
yyp2=yyp1+lambda.*sin(angle_xy_).*sin(angle_z_);
zzp2=zzp1+lambda.*cos(angle_z_);

vz=[ions_map(:,3);vac_map0(:,3);vac_map2(:,3)];
vy=[ions_map(:,2);vac_map0(:,2);vac_map2(:,2)];
vx=[ions_map(:,1);vac_map0(:,1);vac_map2(:,1)];

keyboard;
epsilon_=zeros(1,Nions+Nvac0+Nvac2);
epsilon_(vz<=layers(1).t,:)=layers(1).k*epsilon0;
epsilon_(vx.^2+vy.^2<((r_GB).^2),:)=kGB*epsilon0;
epsilon_(vz>layers(1).t,:)=layers(2).k*epsilon0;
epsilon=bsxfun(@times,epsilon_,epsilon_')./bsxfun(@plus,epsilon_,epsilon_');
if kBULK~=layers(1).k
    warning('!!! Bulk.k is not equal to passed-in Bulk.k');
end

% Matrices with distances between different ions/vacancies
% dist(i,j) = distance between ions i and j
if VpinChargeFlag==2 %+2 charge when V is off
    if fprintf('
&c Vreram==0, Vpin charge = +2
');
        distx=
            bsxfun(@minus,[ions_map(:,1)',vac_map0(:,1)',vac_map2(:,1)'],[ions_map(:,1);vac_map0(:,1);vac_map2(:,1)]);
        disty=
            bsxfun(@minus,[ions_map(:,2)',vac_map0(:,2)',vac_map2(:,2)'],[ions_map(:,2);vac_map0(:,2);vac_map2(:,2)]);
        distz=
            bsxfun(@minus,[ions_map(:,3)',vac_map0(:,3)',vac_map2(:,3)'],[ions_map(:,3);vac_map0(:,3);vac_map2(:,3)]);
        Q_=
            bsxfun(@times,[Q_i*ones(1,Nions),Q_v*ones(1,Nvac0),Q_v2*ones(1,Nvac2)],[ones(Nions,1);ones(Nvac0,1);ones(Nvac2,1)]);
        dist=sqrt(distx.^2+disty.^2+distz.^2);
    end
if VpinChargeFlag==1 %-1 charge when V is on
    if fprintf('
&c Vreram=>0||<0, Vpin charge = +1
');
        distx=
            bsxfun(@minus,[ions_map(:,1)',vac_map0(:,1)',vac_map2(:,1)'],[ions_map(:,1);vac_map0(:,1);vac_map2(:,1)]);
        disty=
            bsxfun(@minus,[ions_map(:,2)',vac_map0(:,2)',vac_map2(:,2)'],[ions_map(:,2);vac_map0(:,2);vac_map2(:,2)]);
        distz=
            bsxfun(@minus,[ions_map(:,3)',vac_map0(:,3)',vac_map2(:,3)'],[ions_map(:,3);vac_map0(:,3);vac_map2(:,3)]);
        Q_=
            bsxfun(@times,[Q_i*ones(1,Nions),Q_v*ones(1,Nvac0),Q_v1*ones(1,Nvac2)],[ones(Nions,1);ones(Nvac0,1);ones(Nvac2,1)]);
        dist=sqrt(distx.^2+disty.^2+distz.^2);
    end
% Matrices with fields between different ions/vacancies
%  F_i,j = field at ion/vacancy i due to ion j along the direction i j
A=-q.*Q_./(4.*pi.*epsilon.*(dist.^3));
F_x=A.*distx;
F_y=A.*disty;
F_z=A.*distz;

% Arrays with total field at each ion/vacancy due to the Coulomb interaction
% with the other ions/vacancies
M=logical(diag(ones(1,Nions+Nvac0+Nvac2))|dist==0);
F_x (M)=0; F_y (M)=0; F_z (M)=0;
F_X=sum(F_x.,2); F_Y=sum(F_y.,2); F_Z=sum(F_z.,2);

% "Internal" field due to the ion/vac-ion/vac interaction
F_X=reshape(F_X,[1,Nions+Nvac0+Nvac2,1]);
F_Y=reshape(F_Y,[1,Nions+Nvac0+Nvac2,1]);
F_Z=reshape(F_Z,[1,Nions+Nvac0+Nvac2,1]);

% Dot product to project the field along the radial directions
F_int=bsxfun(@times,F_X,(xxp1-xxp2)./lambda) ...
%F_int=0
Field=F_int+F_ext;

Ea_eff=zeros(size(xxp1));
Ea_eff(V)=Ea(V)+Q(V).*((lambda/2).*Field(V));
Ea_eff(Ea_eff<0)=0;

Tlocal=interp3_linear(Tdev.Xmesh,Tdev.Ymesh,Tdev.Zmesh,Tdev.Tmesh,xxp1,yyp1,zzp1,NaN);
DiffR=zeros(size(xxp1));
DiffR(V)=G0_i.*exp(-q.*Ea_eff(V)./(Kb.*(Tlocal(V))));

%Normalization
DiffR=DiffR*6/(length(angle_xy)*length(angle_z));
end
Kinetic Monte Carlo (Numerical method to calculate event probability)

- Rates associated to each possible transition → 1 x N array
  - (N is the number of possible events)
  - $\sum$ Rates → Divides each rate by sum of all rates $\rightarrow P$

- Uses 2 random #s as inputs** → returns
  - Index corresponding to the next event
  - Lifetime of the system in the current status

- Lifetime: time spent by the system in the current status before the event

% Function that randomly determines the next event within the kinetic
% Monte Carlo algorithm. rates is a 1 x N array (N is the number of possible
% events) representing the rates associated to each transition
% The function returns 1) the index corresponding to the next event
% 2) the lifetime of the system in the current status (i.e. the time spent
% by the system in the current status before the event)
% r is an array of 2 numbers randomly generated between 0 and 1

%ref for paper: Gunther Jegert_TED_2011_Monte Carlo Simulation of Leakage
%Currents in TiN/ZrO2/TiN Capacitors

function [nextevent lifetime]=kMC_nextevent(rates,r)
  rates=reshape(rates,1,[]);
  Nevents=length(rates);
  rates_cumsum=cumsum(rates);
  Rtot=rates_cumsum(Nevents);
  rates_cumsum=[0,rates_cumsum];
  lifetime=-log(r(1))/Rtot;
  nextevent=find(rates_cumsum(1:Nevents)<=r(2)*Rtot &
  r(2)*Rtot<r(2)*rates_cumsum(2:Nevents+1));
end

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%---------------------------Printing-Sub-Function-start----------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

% Function that randomly determines the next event within the kinetic
% Monte Carlo algorithm. rates is a 1 x N array (N is the number of possible
% events)
% representing the rates associated to each transition
% The function returns 1) the index corresponding to the next event
% 2) the lifetime of the system in the current status (i.e. the time spent
% by the system in the current status before the event)
% r is an array of 2 numbers randomly generated between 0 and 1

%ref for paper: Gunther Jegert_TED_2011_Monte Carlo Simulation of Leakage
%Currents in TiN/ZrO2/TiN Capacitors
function [ output_args ] = Untitled( input_args )

%UNTITLED Summary of this function goes here
% Detailed explanation goes here

dist = 1e-12:1e-12:2e-9;
lambda=2.5e-10;
m = ... 
    (1) .* (dist <= 3e-10 ) + ... 
    (exp(-dist./(.6*lambda)+2)) .* (dist > 3e-10 ) ;

mx = ... 
    (1) .* (dist <= 2.5e-10 ) + ... 
    (exp(-dist./(.5*lambda)+2)) .* (dist > 2.5e-10 ) ;

plot(dist,m,dist,mx);

end
function 
[] = Copy_of_ALL_IMW_DDAnimation3(InCF, xspacing, yspacing, zspacing, roz, FrameRate, filename, ~, ~, PULSE_OFF, ~, Start_num, ~, ~) 
global X1_D X2_D; 
global Y1_D Y2_D; 
global layers; 
%keyboard; 
nnn = 2; 
if PULSE_OFF == 0; vramp_rate = 1; end; 
vz = linspace(0, sum([layers(:, t)]), 17) * 1e9; 
vv = linspace(0, X2_D, 11) * 1e9; 
vx = linspace(0, Y2_D, 21) * 1e9; 

fid = fopen('FOutput/TT_x_ro_Nv_NMO_NTE_NR_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp-u_L', 'r'); 
F = reshape(fread(fid, inf, 'double'), [], 22); 
fclose(fid); 
if exist('ROutput', 'dir') 
    fid = fopen('ROutput/TT_x_ro_Nv_NMO_NTE_NR_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp-u_L', 'r'); 
    R = reshape(fread(fid, inf, 'double'), [], 22); 
    fclose(fid); 
end 
if exist('SOutput', 'dir') 
    fid = fopen('SOutput/TT_x_ro_Nv_NMO_NTE_NR_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp-u_L', 'r'); 
    S = reshape(fread(fid, inf, 'double'), [], 22); 
    fclose(fid); 
end 
if exist('R2Output', 'dir') 
    fid = fopen('R2Output/TT_x_ro_Nv_NMO_NTE_NR_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp-u_L', 'r'); 
    R2 = reshape(fread(fid, inf, 'double'), [], 22); 
    fclose(fid);
if exist('S2Output', 'dir')
    fid=fopen('S2Output/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L', 'r');
    S2=reshape(fread(fid, inf, 'double'), [], 22);
    fclose(fid);
end

%keyboard;
F=F(2:end,:);

TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L=[F;R;S;R2;S2];
T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L(1,:)=TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L(end+1,:);
Stop_num=length(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L);
VV=[Start_num:1:Stop_num-1 Stop_num-1:1:Stop_num];
FrameRate=length(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L)/tanimation;

Aft_time=0;
Aft_T=0;

compl=0;

Nx=length(xspacing); Ny=length(yspacing); Nz=length(zspacing);
roHfO2=1e5;
roHf=4e-5;

Zc=(zspacing(1:end-1)+zspacing(2:end))/2;
[X3 Y3 Z3]=meshgrid(xspacing,yspacing,Zc);

[nsteps]=numel(VV);
%nsteps=t_length;
writerObj = VideoWriter(filename);
writerObj.Quality=100;

set(gca,'nextplot','replacechildren');
set(gcf,'Renderer','opengl');

scrsz = get(0,'ScreenSize');
set(gcf,'Position',scrsz(1) scrsz(2)/1.1 scrsz(3) scrsz(4)/1.1);

set(writerObj,'FrameRate',FrameRate);
open(writerObj);
nn=1;
for k=1:nsteps

keyboard;
n=VV(k);

clf
hold off;

subplot(3,3,1,'Position',[0.0700 0.715 0.2134 0.24]);
plot(TV(1:length(F)),I(1:length(F)),'LineWidth',4);
hold on
plot((TV(n)),I(n),'LineStyle','none','Marker','o','MarkerFaceColor','red','MarkerSize',12,'Color','blue');

hhg=title(['I=',num2str((I(n)*1e6)),'uA';'t=',num2str(t(n)),'s']);
set(hhg,'Color','b');
set(hhg,'FontSize',16);

subplot(3,3,2,'Position',[0.3708 0.715 0.2134 0.25]);
if n<=length(F)
    fid=fopen(['FOutput/IonMap_n',num2str(n)],'rb');
    ions_map=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);

    fid=fopen(['FOutput/VacMap0_n',num2str(n)],'rb');
    vac_map0=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);

    fid=fopen(['FOutput/VacMap2_n',num2str(n)],'rb');
    vac_map2=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);

    fid=fopen(['FOutput/RecombMap_n',num2str(n)],'rb');
    Recomb_map=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);

end
fid=fopen(['FOutput/SpacerIonMap_n',num2str(n)],'rb'); SpacerIonMap=reshape(fread(fid,inf,'double'),[],3); fclose(fid);

V=repmat(ions_map(:,3)<=layers(1).t,1,3); ions_map_lay1=reshape(ions_map(V),[],3); ions_map_lay2=reshape(ions_map(~V),[],3);

if ~isempty(ions_map_lay1)
plot3(reshape(ions_map_lay1(:,1),1,[])*1e9,reshape(ions_map_lay1(:,2),1,[])*1e9,reshape(ions_map_lay1(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',5);
hold on;
end
if ~isempty(ions_map_lay2)
plot3(reshape(ions_map_lay2(:,1),1,[])*1e9,reshape(ions_map_lay2(:,2),1,[])*1e9,reshape(ions_map_lay2(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor','w','MarkerSize',5);
hold on;
end
if ~isempty(vac_map0)
plot3(reshape(vac_map0(:,1),1,[])*1e9,reshape(vac_map0(:,2),1,[])*1e9,reshape(vac_map0(:,3),1,[])*1e9,'o','MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize',7);
hold on;
end
if ~isempty(SpacerIonMap)
plot3(reshape(SpacerIonMap(:,1),1,[])*1e9,reshape(SpacerIonMap(:,2),1,[])*1e9,reshape(SpacerIonMap(:,3),1,[])*1e9,'o','MarkerFaceColor','cyan','MarkerSize',6,'Color','black');
hold on;
end
Plot_layers(((X2_D-X1_D)*1e9)),(((Y2_D-Y1_D)*1e9)),(((layers(1).t)*1e9)),'','','','','','',[],0.5,0.5)

hyx=title('Side View');
set(hyx,'Color','b');
%p= get(hyx,'Position');
set(hyx,'Position',[-10.6386,-8,10.5]);
set(hyx,'FontSize',25);
xlabel('x(mm)','FontSize',14);
ylabel('y(mm)','FontSize',14);
zlabel('z(mm)','FontSize',14);

%%%% subplot(3,3,3,'Position',[0.6700 0.715 0.2134 0.23]);
%get(gca,'position')
%%%%

xdens3=zeros(size(X3));

xdens3((X3>0&X3<=layers(1).t)=nmap(vac_map0,X3(X3>0&X3<=layers(1).t),Y3(X3>0&X3<=layers(1).t),Z3(Z3>0&Z3=layers(1).t),xspacing,yspacing,zspacing);
xdens3((xdens3>2)=2;
%     hx=title(' log_1_0(Resitivity)','Position',[-8.3310,-10.1,10.0],'Color','b','FontSize',25);
%     set(hx);
%     xlabel('x(nm)','FontSize',14);
%     ylabel('y(nm)','FontSize',14);
%     zlabel('z(nm)','FontSize',14);
%     view(0,-90);

%----------------------------------------------------------------------
%subplot(3,3,6,'Position',[ 0.6700    0.39    0.2134    0.23]);
%     get(gca,'position')
%--

xdens3=zeros(size(X3));

xdens3(Z3>0&Z3<=layers(1).t)=nmap(vac_map0,X3(Z3>0&Z3<=layers(1).t),Y3(Z3>0&Z3<=layers(1).t),Z3(Z3>0&Z3<=layers(1).t),xspacing,yspacing,zspacing);

zdens3(xdens3>2)=2;

roz(Z3>0 & Z3<=layers(1).t)=(roHfO2+1575).*exp(-(2.5-(2-xdens3(Z3>0&Z3<=layers(1).t))).^6)+roHf;

3Dcontourf_imshow(xspacing,yspacing,zspacing,squeeze(log10(roz)),-10e-9,10e-9,11);

caxis([0 5]);

%     hx=title(' log_1_0(Resitivity)','Position',[-8.3310,-10.1,10.0],'Color','b','FontSize',25);
%     set(hx);
%     xlabel('x(nm)','FontSize',14);
%     ylabel('y(nm)','FontSize',14);
%     zlabel('z(nm)','FontSize',14);
%     view(0,-90);

%-----------------

if n<=length(F)
    fid=fopen(['FOutput/Tmap_n',num2str(n)],'rb');
    Tmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz);
    fclose(fid);
end

plot3Dmap(xspacing,yspacing,zspacing(1,11:51),squeeze(Tmap),-10e-9,10e-9);

%Plot_layers((((X2_D-X1_D))),(((Y2_D-Y1_D))),(((layers(1).t))),(((layers(2).t))),'','','','','','',[])

hhb=title('Temperature (K)');
set(hhb,'Color','b');
%get(hh,'Position')
set(hhb,'Position',[-8.3552,-9.8902,10.9]);
set(hhb,'FontSize',25);
caxis([300 1400]);

xlabel('x(nm)','FontSize',14);
ylabel('y(nm)','FontSize',14);
zlabel('z(nm)','FontSize',14);

%----------------------------------------------------------------------

if n<=length(F)
    fid=fopen(['FOutput/Vmap_n',num2str(n)],'rb');
    Vmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz);
    fclose(fid);
end
 [~,~,Ezmesh]=gradient(Vmap,xspacing(2)-xspacing(1),yspacing(2)-
yspacing(1),zspacing(2)-zspacing(1));
plot3Dmap(xspacing,yspacing,zspacing(1,11:51),abs(Ezmesh),'E-Field (V/m)',-
10e-9,10e-9);
    hhsd=title('E-Field (V/m)');
set(hhsd,'Color','b');
%p= get(hh,'Position')
set(hhsd,'Position',[-4.1776,-4.9451,10.1]);
set(hhsd,'FontSize',25);

    caxis([0 1e9]);
xlabel('x(nm)','FontSize',14);
ylabel('y(nm)','FontSize',14);
zlabel('z(nm)','FontSize',14);

    subplot(3,3,9,'Position',[ 0.6700 0.07 0.16 0.23]);
%get(gca,'position')

%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%---------------------------------Efield 2-D slices--------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

plot(n,NSp(n),'LineStyle','none','Marker','o','MarkerFaceColor','blue','MarkerSize',11,'Color','black');
    % hold on;
    %figure;

    xdens3=zeros(size(X3));

    xdens3 (Z3>0&Z3<=layers(1).t)=nmap(vac_map0,X3(Z3>0&Z3<=layers(1).t),Y3(Z3>0&Z3<=layers(1).t),Z3(Z3>0&Z3<=layers(1).t),xspacing,yspacing,zspacing);
    xdens3 (xdens3>2)=2;
    roz(Z3>0 & Z3<=layers(1).t)=(roHfO2+1575).*exp(-(2.5-(2-
    xdens3(Z3>0&Z3<=layers(1).t))).^6)+roHf;
    x3Dcontourf_imshow(xspacing,yspacing,zspacing,squeeze(log10(roz)),-10e-
    9,10e-9,13);
    caxis([0 5]);
    %hx=title(' log_1_0(Resitivity)','Position',[-8.3310,-
    10.1,10.0],'Color','b','FontSize',25);
    % set(hx);
    xlabel('x(nm)','FontSize',14);
    ylabel('y(nm)','FontSize',14);
    zlabel('z(nm)','FontSize',14);
    view(0,-90);
    set(gcf,'PaperUnits','inches','PaperPosition',[0 0 15 9])

    opengl('software');
drawnow;

    writeVideo(writerObj,getframe(gcf));

    print(gcf,'-dpng',[
    'Transient/maps_n',num2str(n),'.png']);
nn=nn+1;
end
close(writerObj);

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
function
[] = Copy_of_ALL_IMW_DDanimation3(InCF, xspacing, yspacing, zspacing, roz, FrameRate, filename, ~, ~, PULSE_OFF, ~, Start_num, ~, ~)
global X1_D X2_D;
global Y1_D Y2_D;
global layers;
% keyboard;
nnn = 2;
if PULSE_OFF == 0; vramp_rate = 1; end;
vz = linspace(0, sum(layers(:, t)), 17) * 1e9;
vx = linspace(0, X2_D, 11) * 1e9;
vy = linspace(0, Y2_D, 21) * 1e9;

fid = fopen('FOutput/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_V1_TV_VsR_VsL_Rr_NC_Cp_u_L', 'r');
F = reshape(fread(fid, inf, 'double'), [], 22);
fclose(fid);
if exist('ROutput', 'dir')
    fid = fopen('ROoutput/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_V1_TV_VsR_VsL_Rr_NC_Cp_u_L', 'r');
    R = reshape(fread(fid, inf, 'double'), [], 22);
    fclose(fid);
end
if exist('SOutput', 'dir')
    fid = fopen('SOutput/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_V1_TV_VsR_VsL_Rr_NC_Cp_u_L', 'r');
    S = reshape(fread(fid, inf, 'double'), [], 22);
    fclose(fid);
end
if exist('R2Output', 'dir')
    fid = fopen('R2Output/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_V1_TV_VsR_VsL_Rr_NC_Cp_u_L', 'r');
    R2 = reshape(fread(fid, inf, 'double'), [], 22);
    fclose(fid);
end
if exist('S2Output','dir')
  fid=fopen('S2Output/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
  S2=reshape(fread(fid,inf,'double'),[],22);
  fclose(fid);
end

%keyboard;
F=F(2:end,:);
TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L=[F;R;S;R2;S2];
T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc=[F];

%keyboard;
TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L(end+1,1)=TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L(end,1);
Stop_num=length(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc);
VV=[Start_num:1:Stop_num-1 Stop_num-1:1:Stop_num];
VV=[Start_num+10:5000:27828 27829:10:27988 27989:100:Stop_num-1 Stop_num-1:1:Stop_num];
tanimation=100;
FrameRate=length(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc)/(tanimation);

keyboard;

F[T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,1)];
I=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,12);
Vreram=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,13);
Rreram=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,18);
NR=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,7);
NSp=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,9);
Vl=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,14);
Vr=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,13);
TV=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,15);
InCF=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,19);
Rload=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,22);
Aft_time=0;
Aft_T=0;
NG=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,8);
NMO=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,5);
NTE=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,6);

keyboard;

compl=0;
 Nx=length(xspacing); Ny=length(yspacing); Nz=length(zspacing);
\[ Z_c = \frac{z_{\text{spacing}(1:end-1)} + z_{\text{spacing}(2:end)}}{2}; \]
\[ [X_3, Y_3, Z_3] = \text{meshgrid}(x_{\text{spacing}}, y_{\text{spacing}}, Z_c); \]

\[ [n_{\text{steps}}] = \text{numel}(\text{VV}); \]
\[ %[n_{\text{steps}}] = t_{\text{length}}; \]
\[ \text{writerObj} = \text{VideoWriter}(\text{filename}); \]
\[ \text{writerObj}.\text{Quality} = 100; \]

\[ \text{set(gca,'nextplot','replacechildren');} \]
\[ \text{set(gcf,'Renderer','opengl');} \]

\[ \text{scrsz} = \text{get(0,'ScreenSize');} \]
\[ \text{set(gcf,'Position',[scrsz(1) scrsz(2)/1.1 scrsz(3) scrsz(4)/1.1]);} \]

\[ \text{set(writerObj,'FrameRate',FrameRate);} \]
\[ \text{open(writerObj);} \]
\[ \text{nn} = 1; \]
\[ \text{for } k = 1:n_{\text{steps}} \]
    \[ \%\text{keyboard;} \]
    \[ n = \text{VV}(k); \]
    \[ \text{clf} \]
    \[ \text{hold off;} \]
    \[ %----------------------------------------------- \]
    \[ \text{subplot}(3,3,1,'Position', [.0700 0.715 0.2134 0.24]); \]
    \[ %----------------- \]
    \[ \text{plot}(T_1(1:length(F)),I(1:length(F)),'LineWidth',4); \]
    \[ \text{hold on} \]
    \[ \text{plot}((T_1(n)),I(n),'LineStyle','none','Marker','o','MarkerFaceColor','red','MarkerSize',12,'Color','blue'); \]
    \[ \text{hhg} = \text{title}(['I=',num2str(I(n)*1e6),'uA';'t=',num2str(t(n)),'s']); \]
    \[ \text{set(hhg,'Color','b');} \]
    \[ \text{set(hhg,'FontSize',16);} \]

\[ %----------------------------------------------- \]
\[ \text{subplot}(3,3,2,'Position', [.3708 0.715 0.2134 0.25]); \]
\[ %get(gca,'position') \]
\[ %----------------------------------------------- \]
\[ \text{if } n < \text{length}(F)} \]
\[ \text{fid} = \text{fopen}(['FOutput/IonMap_n',num2str(n)],'rb'); \]
\[ \text{ions\_map} = \text{reshape} (\text{fread}(\text{fid},\text{inf},'double'),[],3); \]
\[ \text{fclose(fid);} \]
\[ \text{fid} = \text{fopen}(['FOutput/VacMap0_n',num2str(n)],'rb'); \]
\[ \text{vac\_map0} = \text{reshape} (\text{fread}(\text{fid},\text{inf},'double'),[],3); \]
\[ \text{fclose(fid);} \]
\[ \text{fid} = \text{fopen}(['FOutput/VacMap2_n',num2str(n)],'rb'); \]
\[ \text{vac\_map2} = \text{reshape} (\text{fread}(\text{fid},\text{inf},'double'),[],3); \]
\[ \text{fclose(fid);} \]
\[ \text{fid} = \text{fopen}(['FOutput/RecombMap_n',num2str(n)],'rb'); \]
\[ \text{Recomb\_map} = \text{reshape} (\text{fread}(\text{fid},\text{inf},'double'),[],3); \]
\[ \text{fclose(fid);} \]
%----------------
fid=fopen(['FOutput/Vmap_n',num2str(n)],'rb');
Vmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz);
close(fid);

[~,~,Ezmesh]=gradient(Vmap,xspacing(2)-xspacing(1),yspacing(2)-
yspacing(1),zspacing(2)-zspacing(1));
x3Dcontourf_imshow(xspacing,yspacing,zspacing,abs(Ezmesh),-10e-9,10e-9,11);
caxis([0 1e9]);

%     hx=title(' log_1_0(Resitivity)','Position',[-8.3310,-
10.1,10.0],'Color','b','FontSize',25);
%     set(hx);
xlabel('x(nm)','FontSize',14);
ylabel('y(nm)','FontSize',14);
zlabel('z(nm)','FontSize',14);
view(0,-90);
cf,'PaperUnits','inches','PaperPosition',[0 0 15 9])

%----------------
end

%----------------
if n<=length(F)
    fid=fopen(['FOutput/Tmap_n',num2str(n)],'rb');
    Tmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz);
close(fid);
end

plot3Dmap(xspacing,yspacing,zspacing(1,11:51),squeeze(Tmap),-10e-9,10e-9);

%Plot_layers(((X2_D-X1_D)),(((Y2_D-
Y1_D))),(((layers(1).t))),(((layers(2).t))),'','','','','','',
[]) hhb=title(' Temperature (K)');
set(hhb,'Color','b');
set(hhb,'Position',[-8.3552,-9.8902,10.9]);
set(hhb,'FontSize',25);
xaxis([300 1400]);
ylabel('y(nm)','FontSize',14);
zlabel('z(nm)','FontSize',14);

%----------------
end

%----------------
if n<=length(F)
    fid=fopen(['FOutput/Vmap_n',num2str(n)],'rb');
    Vmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz);
close(fid);
end

-%- %--------------------
[-,~,Ezmesh]=gradient(Vmap,xspacing(2)-xspacing(1),yspacing(2)-
yspacing(1),zspacing(2)-zspacing(1));
plot3Dmap(xspacing,yspacing,zspacing(1,11:51),abs(Ezmesh),'E-Field (V/m)',-10e-9,10e-9);

hhad=title('E-Field (V/m)');
set(hhad,'Color','b');
set(hhad,'Position',[-4.1776,-4.9451,10.1]);
set(hhad,'FontSize',25);
xlabel('x(nm)','FontSize',14);
ylabel('y(nm)','FontSize',14);
zdens3=zeros(size(X3));

xdens3(Z3>0 & Z3<=layers(1).t)=nmap(vac_map0,X3(Z3>0 & Z3<=layers(1).t),Y3(Z3>0 & Z3<=layers(1).t),Z3(Z3>0 & Z3<=layers(1).t),xspacing,yspacing,zspacing);
xdens3(xdens3>2)=2;
roz(Z3>0 & Z3<=layers(1).t)=(roHfO2+1575).*exp(-(2.5-(2-
xdens3(Z3>0 & Z3<=layers(1).t))).^6)+roHf;

x3Dcontourf_imshow(xspacing,yspacing,zspacing,squeeze(log10(roz)),-10e-
9,10e-9,13);
caxis([0 5]);

%---------------------------------V Potential 2-D slices---------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

%---------------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------

255
function

```matlab
global X1_D X2_D;
global Y1_D Y2_D;
global layers;
%keyboard;
nnn=2;
if PULSE_OFF==0; vramp_rate=1; end;
vz=linspace(0,sum([layers(:,t)]),17)*1e9;
vx=linspace(0,X2_D,11)*1e9;
vy=linspace(0,Y2_D,21)*1e9;

fid=fopen('FOutput/TT_x_ro_Nv_NMO_NTE_NR_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
F=reshape(fread(fid,inf,'double'),[],22);
fclose(fid);
if exist('ROutput','dir')
    fid=fopen('ROutput/TT_x_ro_Nv_NMO_NTE_NR_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
    R=reshape(fread(fid,inf,'double'),[],22);
fclose(fid);
end
if exist('SOutput','dir')
    fid=fopen('SOutput/TT_x_ro_Nv_NMO_NTE_NR_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
    S=reshape(fread(fid,inf,'double'),[],22);
fclose(fid);
end
if exist('R2Output','dir')
    fid=fopen('R2Output/TT_x_ro_Nv_NMO_NTE_NR_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
    R2=reshape(fread(fid,inf,'double'),[],22);
fclose(fid);
end
```
if exist('S2Output', 'dir')
    fid=fopen('S2Output/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L', 'r');
    S2=reshape(fread(fid, inf, 'double'), [], 22);
    fclose(fid);
end

%keyboard;
F=F(2:end,:);
TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L=[F;R;S;R2;S2];
T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc=[F];
%keyboard;
TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L(end+1, 1)=TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L(end, 1);
Stop_num=length(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc);
VV=[Start_num:1:Stop_num-1 Stop_num-1:1:Stop_num];
VV=[Start_num+10:5000:27828 27829:10:27988 27989:100:Stop_num-1 Stop_num-1:1:Stop_num];
FrameRate=length(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc) / (tanimation);

keyboard;

F=

[T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 1),]
I=[T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 12),]
Vreram=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 13);
Rreram=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 18);
NR=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 7);
NSp=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 9);
Vl=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 14);
Vr=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 13);
TV=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 15);
InCF=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 19);
Rload=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 22);
Aft_time=0;
Aft_T=0;
NG=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 8);
NMO=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 5);
NTE=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 6);

keyboard;
compl=0;
Nx=length(xspacing); Ny=length(yspacing); Nz=length(zspacing);
roHfO2=1e5;
roHf=4e-5;
Zc=(zspacing(1:end-1)+zspacing(2:end))/2;
[X3 Y3 Z3]=meshgrid(xspacing,yspacing,Zc);

[nsteps]=numel(VV);
%[nsteps]=t_length;
writerObj = VideoWriter(filename);
writerObj.Quality=100;

set(gca,'nextplot','replacechildren');
set(gcf,'Renderer','opengl');

scrsz = get(0,'ScreenSize');
set(gcf,'Position',[scrsz(1) scrsz(2)/1.1 scrsz(3) scrsz(4)/1.1]);

set(writerObj,'FrameRate',FrameRate);
open(writerObj);
nn=1;
for k=1:nsteps

keyboard;
n=VV(k);
clf
hold off;

subplot(3,3,1,'Position',[0.0700 0.715 0.2134 0.24]);
-----------------
plot(TV(1:length(F)),I(1:length(F)),'LineWidth',4);
hold on
plot((TV(n)),I(n),'LineStyle','none','Marker','o','MarkerFaceColor','red','MarkerSize',12,'Color','blue');

hhg=title(['                             I=',num2str((I(n)*1e6)),'uA',';
                    t=',num2str(t(n)),'s']);
set(hhg,'Color','b');
set(hhg,'FontSize',16);
-----------------

subplot(3,3,2,'Position',[ 0.3708 0.715 0.2134 0.25]);

if n<=length(F)
    fid=fopen(['FOutput/IonMap_n',num2str(n)],'rb');
    ions_map=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);

    fid=fopen(['FOutput/VacMap0_n',num2str(n)],'rb');
    vac_map0=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);

    fid=fopen(['FOutput/VacMap2_n',num2str(n)],'rb');
    vac_map2=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);

    fid=fopen(['FOutput/RecombMap_n',num2str(n)],'rb');
    Recomb_map=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);
end

plot((TV(n)),I(n),'LineStyle','none','Marker','o','MarkerFaceColor','red','MarkerSize',12,'Color','blue');

hhg=title(['                             I=',num2str((I(n)*1e6)),'uA',';
                    t=',num2str(t(n)),'s']);
set(hhg,'Color','b');
set(hhg,'FontSize',16);

end
fid=fopen(['FOutput/SpacerIonMap_n',num2str(n)],'rb'); SpacerIonMap=reshape(fread(fid,inf,'double'),[],3); fclose(fid);
end

V=repmat(ions_map(:,3)<=layers(1).t,1,3);
ions_map_lay1=reshape(ions_map(V),[],3);
ions_map_lay2=reshape(ions_map(~V),[],3);

if ~isempty(ions_map_lay1)
    plot3(reshape(ions_map_lay1(:,1),1,[])*1e9,reshape(ions_map_lay1(:,2),1,[])*1e9,
         reshape(ions_map_lay1(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',5);
    hold on;
end

if ~isempty(ions_map_lay2)
    plot3(reshape(ions_map_lay2(:,1),1,[])*1e9,reshape(ions_map_lay2(:,2),1,[])*1e9,
         reshape(ions_map_lay2(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor','w','MarkerSize',5);
    hold on;
end

if ~isempty(vac_map0)
    plot3(reshape(vac_map0(:,1),1,[])*1e9,reshape(vac_map0(:,2),1,[])*1e9,reshape(vac_map0(:,3),1,[])*1e9,'o','MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize',7);
    hold on;
end

if ~isempty(SpacerIonMap)
    plot3(reshape(SpacerIonMap(:,1),1,[])*1e9,reshape(SpacerIonMap(:,2),1,[])*1e9,reshape(SpacerIonMap(:,3),1,[])*1e9,'o','MarkerFaceColor','cyan','MarkerSize',6,'Color','black');
    hold on;
end

Plot_layers(((X2_D-X1_D)*1e9),(((Y2_D-Y1_D)*1e9)),(((layers(1).t)*1e9)),(((layers(2).t)*1e9)),'','','','','','',[],.05,.5)

hyx=title('Side View');
sel(hyx,'Color','b');
%p= get(hyx,'Position');
set(hyx,'FontSize',25);
xlabel('x(nm)', 'FontSize', 14);
ylabel('y(nm)', 'FontSize', 14);
zlabel('z(nm)', 'FontSize', 14);

%-----------------
subplot(3,3,3, 'Position', [0.6700 0.715 0.2134 0.23]);
%get(gca, 'position')
%-----------------
fid=fopen(['FOutput/Vmap_n',num2str(n)],'rb'); Vmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz); fclose(fid);
[~,~,Ezmesh]=gradient(Vmap,xspacing(2)-xspacing(1),yspace(2)-yspace(1),zspace(2)-zspace(1));
x3Dcontourf_imshow(xspacing, yspacing, zspacing, abs(Vmap), -10e-9, 10e-9, 33);
caxis([0 1.8]);
xlabel('x(nm)', 'FontSize', 14);
ylabel('y(nm)', 'FontSize', 14);
zlabel('z(nm)', 'FontSize', 14);
view(0, -90);

hxx = title('Bottom View');
set(hxx, 'Color', 'b');
%p = get(hxx, 'Position');
set(hxx, 'Position', [-0.0244, -10.9, -1.0005]);
set(hxx, 'FontSize', 25);
xlabel('x(nm)', 'FontSize', 14);
ylabel('y(nm)', 'FontSize', 14);
zlabel('z(nm)', 'FontSize', 14);

%-----------------
subplot(3, 3, 4, 'Position', [0.0700 0.39 0.20 0.25]);
%get(gca,'position')
%-----------------
ions_map = ions_map_lay1;

xdens3 = zeros(size(X3));

xdens3(Z3 > 0 & Z3 <= layers(1).t) = nmap(vac_map0, X3(Z3 > 0 & Z3 <= layers(1).t), Y3(Z3 > 0 & Z3 <= layers(1).t), xspacing, yspacing, zspacing);

xdens3(xdens3 > 2) = 2;

roz(Z3 > 0 & Z3 <= layers(1).t) = (roHfO2 + 1575) .* exp(-(2.5 - (2 - xdens3(Z3 > 0 & Z3 <= layers(1).t)).^6)) + roHf;

plot3Dmap(xspacing, yspacing, zspacing(1, 11:51), squeeze(log10(roz)), -10e-9, 10e-9);
caxis([0 5]);
hx = title('log_10(Resitivity)', 'Position', [-8.3310, -10.1, 10.0], 'Color', 'b', 'FontSize', 25);
set(hx);
xlabel('x(nm)', 'FontSize', 14);
ylabel('y(nm)', 'FontSize', 14);
zlabel('z(nm)', 'FontSize', 14);

%---------------------------------------------------------------------
subplot(3, 3, 5, 'Position', [0.3708 0.39 0.2134 0.23]);
%get(gca,'position')
%-----------------
fid = fopen(['FOutput/Vmap_n', num2str(n)], 'rb');
Vmap = reshape(fread(fid, Nx*Ny*Nz, 'double'), Nx, Ny, Nz);
caxis([0 1.8]);

%x3Dcontourf_imshow(xspacing, yspacing, zspacing(abs(Vmap), -10e-9, 10e-9, 12);
caxis([0 1.8]);

% hx = title('log_10(Resitivity)', 'Position', [-8.3310, -10.1, 10.0], 'Color', 'b', 'FontSize', 25);
% set(hx);
xlabel('x(nm)', 'FontSize', 14);
ylabel('y(nm)', 'FontSize', 14);
zlabel('z(nm)', 'FontSize', 14);
view(0, -90);

%---------------------------------------------------------------------
subplot(3, 3, 6, 'Position', [0.6700 0.39 0.2134 0.23]);
%get(gca,'position')
%--
%fid = fopen(['FOutput/Vmap_n', num2str(n)], 'rb');
Vmap = reshape(fread(fid, Nx*Ny*Nz, 'double'), Nx, Ny, Nz);
caxis([0 1.8]);

% hx = title('log_10(Resitivity)', 'Position', [-8.3310, -10.1, 10.0], 'Color', 'b', 'FontSize', 25);
% set(hx);
xlabel('x(nm)', 'FontSize', 14);
ylabel('y(nm)', 'FontSize', 14);
zlabel('z(nm)', 'FontSize', 14);
view(0, -90);
[~,~,Ezmesh]=gradient(Vmap,xspacing(2)-xspacing(1),yspacing(2)-yspacing(1),zspacing(2)-zspacing(1));
x3Dcontourf.imshow(xspacing,yspacing,zspacing,abs(Vmap),-10e-9,10e-9,11);
caxis([0 1.8]);
xlabel('x(nm)', 'FontSize', 14);
ylabel('y(nm)', 'FontSize', 14);
zlabel('z(nm)', 'FontSize', 14);
view(0,-90);

%-----------------
subplot(3,3,7,'Position',[0.0700 0.05 0.20 0.23]);
%get(gca,'position')
%-----------------
if n<=length(F)
    fid=fopen(['FOutput/Tmap_n',num2str(n)],'rb');
    Tmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz);
    fclose(fid);
end

plot3Dmap(xspacing,yspacing,zspacing(1,11:51),squeeze(Tmap),-10e-9,10e-9);

% Plot_layers((((X2_D-X1_D))),(((Y2_D-Y1_D))),( ((layers(1).t))),((layers(2).t)),'','','','','','',[])

hhb=title('Temperature (K)');
set(hhb,'Color','b');
%p=get(hh,'Position')
set(hhb,'Position',[-8.3552,-9.8902,10.9]);
set(hhb,'FontSize',25);
caxis([300 1400]);
xlabel('x(nm)', 'FontSize', 14);
ylabel('y(nm)', 'FontSize', 14);
zlabel('z(nm)', 'FontSize', 14);
%-----------------
subplot(3,3,8,'Position',[0.3708 0.05 0.20 0.23]);
% get(gca,'position')
%-----------------

if n<=length(F)
    fid=fopen(['FOutput/Vmap_n',num2str(n)],'rb');
    Vmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz);
    fclose(fid);
end

[~,~,Ezmesh]=gradient(Vmap,xspacing(2)-xspacing(1),yspacing(2)-yspacing(1),zspacing(2)-zspacing(1));
plot3Dmap(xspacing,yspacing,zspacing(1,11:51),abs(Vmap),'E-Field (V/m)',-10e-9,10e-9);

hhbsd=title('Potential (V)');
set(hhbsd,'Color','b');
%p=get(hh,'Position')
set(hhbsd,'Position',[-4.1776,-4.9451,10.1]);
set(hhbsd,'FontSize',25);
caxis([0 1.8]);
xlabel('x(nm)', 'FontSize', 14);
ylabel('y(nm)', 'FontSize', 14);
zlabel('z(nm)', 'FontSize', 14);
%-----------------
 subplot(3,3,9,'Position',[0.6700 0.07 0.16 0.23]);
 %get(gca,'position')
%-----------------
xdens3=zeros(size(X3));
xdens3(Z3>0&Z3<=layers(1).t)=nmap(vac_map0,X3(Z3>0&Z3<=layers(1).t),Y3(Z3>0&Z3<=layers(1).t),xspacing,yspacing,zspacing);
xdens3(xdens3>2)=2;
roz(Z3>0 & Z3<=layers(1).t)=(roHfO2+1575).*exp(-(2.5-(2-
xdens3(2>0&Z3<=layers(1).t))).^6)+roHf;
x3Dcontourf_imshow(xspacing,yspacing,zspacing,squeeze(log10(roz)),-10e-
9,10e-9,13);
caxis([0 5]);
%    hx=title(' log_1_0(Resitivity)','Position',
[-8.3310,-
10.1,10.0],'Color','b','FontSize',25);
%    set(hx);
    xlabel('x(nm)','FontSize',14);
    ylabel('y(nm)','FontSize',14);
    zlabel('z(nm)','FontSize',14);
    view(0,-90);

set(gcf,'PaperUnits','inches','PaperPosition',[0 0 15 9])
openl('software');
drawnow;
writeVideo(writerObj,getframe(gcf));
print(gcf,'-dpng',['Transient/maps_n',num2str(n),'.png']);
nn=nn+1;
end
close(writerObj);
end

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%---------------------------------Temperature 2-D slices---------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
function

function = Copy_of_ALL_IMW_DDanimation3(InCF, xspacing, yspacing, zspacing, roz, FrameRate, filename, ~, ~, ~, PULSE_OFF, ~, Start_num, ~, ~)

% keyboard;

nnn=2;

if PULSE_OFF==0; vramp_rate=1; end;

if exist('ROutput','dir')
    fid=fopen('ROutput/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
    R=reshape(fread(fid,inf,'double'),[],22);
    fclose(fid);
end

if exist('SOutput','dir')
    fid=fopen('SOutput/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
    S=reshape(fread(fid,inf,'double'),[],22);
    fclose(fid);
end

if exist('R2Output','dir')
    fid=fopen('R2Output/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
    R2=reshape(fread(fid,inf,'double'),[],22);
    fclose(fid);
end

if exist('S2Output','dir')
    fid=fopen('S2Output/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
    S2=reshape(fread(fid,inf,'double'),[],22);
    fclose(fid);
end

% keyboard;

X1_D X2_D;

global Y1_D Y2_D;

layers;

X2_D*1e9;

vy=linspace(0,Y2_D,21)*1e9;

fid=fopen('FOutput/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
F=reshape(fread(fid,inf,'double'),[],22);
fclose(fid);
fid=fopen('S2Output/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
S2=reshape(fread(fid,inf,'double'),[],22);
fclose(fid);
end
%keyboard;
%F=F(2:end,:);
%TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L=[F;R;S;R2;S2];
T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc=[F];
%keyboard;
%TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L(end+1,1)=TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L(end,1);
Stop_num=length(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc);
%keyboard;
%VV=[Start_num:Stop_num-1 Stop_num-1:1:Stop_num];
%VV=[Start_num:10:27828 27828:10:27989 27989:100:Stop_num-1 Stop_num-1:1:Stop_num];
tanimation=100;
FrameRate=length(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc)/(tanimation);
%keyboard;
t=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,1);
I=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,12);
Vreram=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,13);
Rreram=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,18);
NR=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,7);
NSp=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,9);
Vl=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,14);
Vr=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,13);
TV=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,15);
InCF=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,19);
Rload=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,22);
Aft_time=0;
Aft_T=0;
NG=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,8);
NMO=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,5);
NTE=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,6);
%keyboard;
compl=0;
Nx=length(xspacing);Ny=length(yspacing);Nz=length(zspacing);
roHfO2=1e5;
\[ \text{roHf} = 4e-5; \]
\[ Zc = (zspacing(1:end-1)+zspacing(2:end))/2; \]
\[ [X3 Y3 Z3] = \text{meshgrid}(xspacing, yspacing, Zc); \]
\[ [\text{nsteps}] = \text{numel}(VV); \]
\[ %[\text{nsteps}] = \text{t_length}; \]
\[ \text{writerObj} = \text{VideoWriter}(\text{filename}); \]
\[ \text{writerObj.Quality} = 100; \]
\[ \text{set(gca,'nextplot','replacechildren');} \]
\[ \text{set(gcf,'Renderer','opengl');} \]
\[ \text{scrsz} = \text{get(0,'ScreenSize');} \]
\[ \text{set(gcf,'Position',[]} \]
\[ \text{set(writerObj,'FrameRate',FrameRate);} \]
\[ \text{open(writerObj);} \]
\[ \text{nn} = 1; \]
\[ \text{for k} = 1:\text{nsteps} \]
\[ \quad \text{keyboard;} \]
\[ \quad \text{n} = VV(k); \]
\[ \quad \text{clf} \]
\[ \quad \text{hold off;} \]
\[ \quad \text{-----------------------------------------------} \]
\[ \quad \text{subplot}(3,3,1,'Position',[]} \]
\[ \quad \text{plot}(TV(1:length(F)),I(1:length(F)),'LineWidth',4); \]
\[ \quad \text{hold on} \]
\[ \quad \text{plot}(TV(n),I(n),'LineStyle','none','Marker','o','MarkerFaceColor','red','MarkerSize',12,'Color','blue'); \]
\[ \quad \text{hhg} = \text{title(['} \]
\[ \quad \quad \text{I'} \]
\[ \quad \quad = \text{num2str((I(n)*1e6))},'uA',';} \]
\[ \quad \quad \text{t'} \]
\[ \quad \quad = \text{num2str(t(n)),'} \]
\[ \quad \quad \text{s'} \]
\[ \quad \quad ]); \]
\[ \quad \text{set(hhg,'Color','b');} \]
\[ \quad \text{set(hhg,'FontSize',16);} \]
\[ \quad \text{-----------------------------------------------} \]
\[ \quad \text{subplot}(3,3,2,'Position',[]} \]
\[ \quad \text{get(gca,'position')} \]
\[ \quad \text{-----------------------------------------------} \]
\[ \quad \text{if n} < \text{length(F)} \]
\[ \quad \text{fid} = \text{fopen(['} \]
\[ \quad \quad \text{FOutput/IonMap_n',num2str(n)]},'rb');} \]
\[ \quad \text{ions_map} = \text{reshape(fread(fid,inf,'double'),[],3);} \]
\[ \quad \text{fclose(fid);} \]
\[ \quad \text{fid} = \text{fopen(['} \]
\[ \quad \quad \text{FOutput/VacMap0_n',num2str(n)]},'rb');} \]
\[ \quad \text{vac_map0} = \text{reshape(fread(fid,inf,'double'),[],3);} \]
\[ \quad \text{fclose(fid);} \]
\[ \quad \text{fid} = \text{fopen(['} \]
\[ \quad \quad \text{FOutput/VacMap2_n',num2str(n)]},'rb');} \]
\[ \quad \text{vac_map2} = \text{reshape(fread(fid,inf,'double'),[],3);} \]
\[ \quad \text{fclose(fid);} \]
\[ \quad \text{fid} = \text{fopen(['} \]
\[ \quad \quad \text{FOutput/RecombMap_n',num2str(n)]},'rb');} \]
\[ \quad \text{Recomb_map} = \text{reshape(fread(fid,inf,'double'),[],3);} \]
\[ \quad \text{fclose(fid);} \]
\[ \quad \text{fid} = \text{fopen(['} \]
\[ \quad \quad \text{FOutput/SpacerIonMap_n',num2str(n)]},'rb');} \]
\[ \quad \text{fclose(fid);} \]
\[ \end{verbatim}
SpacerIonMap=reshape(fread(fid,inf,'double'),[],3);
fclose(fid);
end
V=repmat(ions_map(:,3)<=layers(1).t,1,3);
ions_map_lay1=reshape(ions_map(V),[],3);
ions_map_lay2=reshape(ions_map(~V),[],3);
if ~isempty(ions_map_lay1)
    plot3(reshape(ions_map_lay1(:,1),1,[])*1e9,reshape(ions_map_lay1(:,2),1,[])*1e9 ,reshape(ions_map_lay1(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',5);
    hold on;
end
if ~isempty(ions_map_lay2)
    plot3(reshape(ions_map_lay2(:,1),1,[])*1e9,reshape(ions_map_lay2(:,2),1,[])*1e9,reshape(ions_map_lay2(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor','w','MarkerSize',5);
    hold on;
end
if ~isempty(vac_map0)
    plot3(reshape(vac_map0(:,1),1,[])*1e9,reshape(vac_map0(:,2),1,[])*1e9,reshape(vac_map0(:,3),1,[])*1e9,'o','MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize',7);
    hold on;
end
if ~isempty(SpacerIonMap)
    plot3(reshape(SpacerIonMap(:,1),1,[])*1e9,reshape(SpacerIonMap(:,2),1,[])*1e9,reshape(SpacerIonMap(:,3),1,[])*1e9,'o','MarkerFaceColor','cyan','MarkerSize',6,'Color','black');
    hold on;
end
Plot_layers((((X2_D-X1_D)*1e9)),(((Y2_D-Y1_D)*1e9)),(((layers(1).t)*1e9)),(((layers(2).t)*1e9)),'','','','','','',[],.05,.5)

hyx=title('Side View');
set(hyx,'Color','b');
%p= get(hyx,'Position');
set(hyx,'Position',[-10.6386,-8,10.5]);
set(hyx,'FontSize',25);
xlabel('x(nm)','FontSize',14);
ylabel('y(nm)','FontSize',14);
zlabel('z(nm)','FontSize',14);

%-----------------
subplot(3,3,3,'Position',[0.6700 0.715 0.2134 0.23]);
%get(gca,'position')
%-----------------

fid=fopen(['FOutput/Tmap_n',num2str(n)],'rb');
Tmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz);
x3Dcontourf_imshow(xspacing,yspacing,zspacing,squeeze(Tmap),-10e-9,10e-9,33);
caxis([300 1400]);
xlabel('x(nm)', 'FontSize',14);
ylabel('y(nm)', 'FontSize',14);
zlabel('z(nm)', 'FontSize',14);
view(0,-90);
hxx=title('Bottom View');
set(hxx, 'Color', 'b');
% p= get(hxx, 'Position');
set(hxx, 'Position', [-0.0244, -10.9, -1.0005]);
set(hxx, 'FontSize', 25);
xlabel('x(nm)', 'FontSize', 14);
ylabel('y(nm)', 'FontSize', 14);
zlabel('z(nm)', 'FontSize', 14);
%-----------------
subplot(3, 3, 4, 'Position', [0.0700, 0.39, 0.20, 0.25]);
% get(gca, 'position')
%-----------------
ions_map = ions_map_lay1;

xdens3 = zeros(size(X3));

xdens3(Z3 > 0 & Z3 <= layers(1).t) = nmap(vac_map0, X3(Z3 > 0 & Z3 <= layers(1).t), Y3(Z3 > 0 & Z3 <= layers(1).t), Xspacing, Yspacing, Zspacing);

xdens3(xdens3 > 2) = 2;

roz(Z3 > 0 & Z3 <= layers(1).t) = (roHfO2 + 1575) .* exp(-(2.5 - (2 - xdens3(Z3 > 0 & Z3 <= layers(1).t))).^6) + roHf;

plot3Dmap(xspacing, Yspacing, Zspacing(1, 11:51), squeeze(log10(roz)), -10e-9, 10e-9);
caxis([0 5]);
hx = title('log_1_0(Resitivity)', 'Position', [-8.3310, -10.1, 10.0], 'Color', 'b', 'FontSize', 25);
set(hx);

%-----------------------------------------------
subplot(3, 3, 5, 'Position', [0.3708, 0.39, 0.2134, 0.23]);
% get(gca, 'position')
%-----------------------------------------------
 fid = fopen(['FOutput/Tmap_n', num2str(n), 'rb']);
 Tmap = reshape(fread(fid, Nx*Ny*Nz, 'double'), Nx, Ny, Nz);
 x3Dcontourf_imshow(xspacing, Yspacing, Zspacing, squeeze(Tmap), -10e-9, 10e-9, 12);
caxis([300 1400]);

%-----------------------------------------------
subplot(3, 3, 6, 'Position', [0.6700, 0.39, 0.2134, 0.23]);
% get(gca, 'position')
%-----------------------------------------------
 fid = fopen(['FOutput/Tmap_n', num2str(n), 'rb']);
 Tmap = reshape(fread(fid, Nx*Ny*Nz, 'double'), Nx, Ny, Nz);

%-----------------------------------------------
x3Dcontourf_imshow(xspacing,yspacing,zspacing,squeeze(Tmap),-10e-9,10e-9,11);
caxis([300 1400]);
% set(hx);
xlabel('x(nm)','FontSize',14);
ylabel('y(nm)','FontSize',14);
zlabel('z(nm)','FontSize',14);
view(0,-90);
%-----------------
subplot(3,3,7,'Position',[0.0700 0.05 0.20 0.23]);
%get(gca,'position')
%-----------------
if n<=length(F)
    fid=fopen(['FOutput/Tmap_n',num2str(n)],'rb');
    Tmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz);
    fclose(fid);
end
plot3Dmap(xspacing,yspacing,zspacing(1,11:51),squeeze(Tmap),-10e-9,10e-9);
hhb=title('Temperature (K)');
set(hhb,'Color','b');
%p=get(hh,'Position')
set(hhb,'Position',[-8.3552,-9.8902,10.9]);
set(hhb,'FontSize',25);
caxis([300 1400]);
xlabel('x(nm)','FontSize',14);
ylabel('y(nm)','FontSize',14);
zlabel('z(nm)','FontSize',14);
%-----------------
subplot(3,3,8,'Position',[0.3708 0.05 0.20 0.23]);
% get(gca,'position')
%-----------------
if n<=length(F)
    fid=fopen(['FOutput/Vmap_n',num2str(n)],'rb');
    Vmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz);
    fclose(fid);
end
[~,~,Ezmesh]=gradient(Vmap,xspacing(2)-xspacing(1),yspacing(2)-yspacing(1),zspacing(2)-zspacing(1));
plot3Dmap(xspacing,yspacing,zspacing(1,11:51),abs(Ezmesh),'E-Field (V/m)',-10e-9,10e-9);
hhsd=title('E-Field (V/m)');
set(hhsd,'Color','b');
%p= get(hh,'Position')
set(hhsd,'Position',[-4.1776,-4.9451,10.1]);
set(hhsd,'FontSize',25);
caxis([0 1e9]);
xlabel('x(nm)','FontSize',14);
ylabel('y(nm)','FontSize',14);
zlabel('z(nm)','FontSize',14);
%-----------------
subplot(3,3,9,'Position',[0.6700 0.07 0.16 0.23]);
xdens3=zeros(size(X3));
xdens3(X3>0&Z3<=layers(1).t)=nmap(vac_map0,X3(Z3>0&Z3<=layers(1).t),Y3(Z3>0&Z3<=layers(1).t),Z3(Z3>0&Z3<=layers(1).t),xspacing,yspacing,zspacing);
function

N=Normal_ALLPlots(InCF,xspacing,yspacing,zspacing,roz,FrameRate,filename,~,~)

%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%---------------------------------All 3D-------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

function

N=Normal_ALLPlots(InCF,xspacing,yspacing,zspacing,roz,FrameRate,filename,~,~)

%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%---------------------------------All 3D-------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

function

N=Normal_ALLPlots(InCF,xspacing,yspacing,zspacing,roz,FrameRate,filename,~,~)

%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%---------------------------------All 3D-------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

function

N=Normal_ALLPlots(InCF,xspacing,yspacing,zspacing,roz,FrameRate,filename,~,~)

%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%---------------------------------All 3D-------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

function

N=Normal_ALLPlots(InCF,xspacing,yspacing,zspacing,roz,FrameRate,filename,~,~)

%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%---------------------------------All 3D-------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

function

N=Normal_ALLPlots(InCF,xspacing,yspacing,zspacing,roz,FrameRate,filename,~,~)

%----------------------------------------------------------------------------
%---------------------------Sub-Function-start-------------------------------
%----------------------------------------------------------------------------
%---------------------------------All 3D-------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

%----------------------------------------------------------------------------
%----------------------------------------------------------------------------
%----------------------------------------------------------------------------

function

N=Normal_ALLPlots(InCF,xspacing,yspacing,zspacing,roz,FrameRate,filename,~,~)
global layers;
%keyboard;
nnn=2;
if PULSE_OFF==0; vramp_rate=1; end;
vz=linspace(0,sum([layers(:,1)]),17)*1e9;
vx=linspace(0,X2_D,11)*1e9;
vz=linspace(0,Y2_D,21)*1e9;

fid=fopen('FOutput/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
F=reshape(fread(fid,inf,'double'),[],22);
fclose(fid);
if exist('ROutput','dir')
fid=fopen('ROutput/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
R=reshape(fread(fid,inf,'double'),[],22);
fclose(fid);
end
if exist('SOutput','dir')
fid=fopen('SOutput/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
S=reshape(fread(fid,inf,'double'),[],22);
fclose(fid);
end
if exist('R2Output','dir')
fid=fopen('R2Output/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
R2=reshape(fread(fid,inf,'double'),[],22);
fclose(fid);
end
if exist('S2Output','dir')
fid=fopen('S2Output/TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L','r');
S2=reshape(fread(fid,inf,'double'),[],22);
fclose(fid);
end
%keyboard;
%F=F(2:end,:);
%TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L=[F;R;S;R2;S2];
T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc=[F];
%keyboard;
%TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L(end+1,1)=TT_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_L(end,1);
Stop_num=length(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc);
VV=[Start_num:1:Stop_num-1 Stop_num-1:1:Stop_num];
VV=[Start_num:140 141:20:Stop_num-1:1:Stop_num];
FrameRate=length(T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc);
%keyboard;
t=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,1);
I=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,2);
Vreram=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,13);
Rreram=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,18);
NR=T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num,7);
NSp = T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 9);
Vl = T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 14);
Vr = T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 13);
TV = T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 15);
InCF = T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 19);
Rload = T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 22);
Aft_time = 0;
Aft_T = 0;
NG = T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rs_NC_Cp_u_Rc(Start_num:Stop_num, 8);
NMO = T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 5);
NTE = T_x_ro_Nv_NMO_NTE_NR_NG_NS_T_E_I_Vr_Vl_TV_VsR_VsL_Rr_NC_Cp_u_Rc(Start_num:Stop_num, 6);
% keyboard;
compl = 0;

Nx = length(xspacing); Ny = length(yspacing); Nz = length(zspacing);
roHfO2 = 1e5;
roHf = 4e-5;

Zc = (zspacing(1:end-1) + zspacing(2:end))/2;
[X3 Y3 Z3] = meshgrid(xspacing, yspacing, Zc);

[nsteps] = numel(VV);
% [nsteps] = t_length;
writerObj = VideoWriter(filename);
writerObj.Quality = 100;

set(gca, 'nextplot', 'replacechildren');
set(gcf, 'Renderer', 'opengl');
scrsz = get(0, 'ScreenSize');
set(gcf, 'Position', [scrsz(1) scrsz(2)/1.1 scrsz(3) scrsz(4)/1.1]);

set(writerObj, 'FrameRate', FrameRate);
open(writerObj);
nn = 1;
for k = 1:nsteps
    % keyboard;
    n = VV(k);
    clf
    hold off;
    subplot(3, 3, 1, 'Position', [0.0700 0.715 0.2134 0.24]);
    plot(TV(1:length(F)), I(1:length(F)), 'LineWidth', 4);
    hold on
plot((TV(n)),I(n),'LineStyle','none','Marker','o','MarkerFaceColor','red','MarkerSize',12,'Color','blue');

hhg=title(
[I=',num2str((I(n)*1e6)),'uA','];
t=',num2str(t(n)),'s']);
set(hhg,'Color','b');
set(hhg,'FontSize',16);

%---------------------------------------------------------------------
subplot(3,3,2,'Position',[ 0.3708 0.715 0.2134 0.25]);
%get(gca,'position')
%-----------------
if n<=length(F)
    fid=fopen(['FOutput/IonMap_n',num2str(n)],'rb');
    ions_map=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);

    fid=fopen(['FOutput/VacMap0_n',num2str(n)],'rb');
    vac_map0=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);

    fid=fopen(['FOutput/VacMap2_n',num2str(n)],'rb');
    vac_map2=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);

    fid=fopen(['FOutput/RecombMap_n',num2str(n)],'rb');
    Recomb_map=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);

    fid=fopen(['FOutput/SpacerIonMap_n',num2str(n)],'rb');
    SpacerIonMap=reshape(fread(fid,inf,'double'),[],3);
    fclose(fid);
end

V=repmat(ions_map(:,3)<=layers(1).t,1,3);
ions_map_lay1=reshape(ions_map(V),[],3);
ions_map_lay2=reshape(ions_map(~V),[],3);

if ~isempty(ions_map_lay1)
    plot3(reshape(ions_map_lay1(:,1),1,[])*1e9,reshape(ions_map_lay1(:,2),1,[])*1e9
        ,reshape(ions_map_lay1(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor'
        ,'r','MarkerSize',5);
    hold on;
end
if ~isempty(ions_map_lay2)
    plot3(reshape(ions_map_lay2(:,1),1,[])*1e9,reshape(ions_map_lay2(:,2),1,[])*1e9
        ,reshape(ions_map_lay2(:,3),1,[])*1e9,'o','MarkerEdgeColor','k','MarkerFaceColor'
        ,'w','MarkerSize',5);
    hold on ;
end
if ~isempty(vac_map0)
    plot3(reshape(vac_map0(:,1),1,[])*1e9,reshape(vac_map0(:,2),1,[])*1e9,reshape(vac_map0(:,3,1,[])*1e9,'o','MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize',7);
    hold on ;
end
if ~isempty(SpacerIonMap)
    plot3(reshape(SpacerIonMap(:,1),1,[])*1e9,reshape(SpacerIonMap(:,2),1,[])*1e9,r...
%-----------------
subplot(3,3,4,'Position',[ 0.0700    0.39    0.20    0.25]);
%get(gca,'position')

%-----------------
% ions_map=ions_map_lay1;

xdens3=zeros(size(X3));

xdens3 (Z3>0&Z3<=layers(1).t)=nmap(vac_map0,X3(Z3>0&Z3<=layers(1).t),Y3(Z3>0&Z3<=layers(1).t),xspacing,yspacing,zspacing);

xdens3 (xdens3>2)=2;

roz(Z3>0 & Z3<=layers(1).t)=(roHfO2+1575).*exp(-(2.5-(2-xdens3 (Z3>0&Z3<=layers(1).t))).^6)+roHf;

plot3Dmap(xspacing,yspacing,zspacing(1,11:51),squeeze(log10(roz)),-10e-9,10e-9);
caxis([0 5]);

hx=title(' log_1_0(Resitivity)', 'Position',[-8.3310,-10.1,10.0], 'Color', 'b', 'FontSize',25);

set(hx);

xlabell('x(mm)', 'FontSize',14);
ylabel('y(mm)', 'FontSize',14);
zlabel('z(mm)', 'FontSize',14);

%---------------------------------------------------------------------
subplot(3,3,5,'Position',[0.3708    0.39    0.2134    0.23]);
%get(gca,'position')

%-----------------
ions_map=ions_map_lay1;

[ITheta, IRrho, IZ]=cart2pol(ions_map(:,1),ions_map(:,2),ions_map(:,3));
[VTheta, VRrho, VZ]=cart2pol(vac_map0(:,1),vac_map0(:,2),vac_map0(:,3));
[n_zhist_I,~] =hist(IRrho(:)*1e9,vx);
[n_zhist_V,~] =hist(VRrho(:)*1e9,vx);

%Radius=[0 1 2 3 4 5 6 7 8 9 10];
Radius=linspace(0,10,11);
for r=1:length(Radius)-1
    Volume(r)=pi*5*(Radius(r+1)^2-Radius(r)^2);
end

normalize=[0 Volume];
n_zhist_V=(n_zhist_V./normalize);
n_zhist_I=(n_zhist_I./normalize);

plot(vx,n_zhist_V,'-b*',vx,n_zhist_I,'-m','LineWidth',3);

hh=title('Radial Distribution');
set(hh,'Color','b');
%p= get(h,'Position')
set(hh,'Position',[4.0,3.85,.6]);
set(hh,'FontSize',25);

set(gca, 'XMinorGrid','on')
xlabel('r (nm)','FontSize',18);
ylabel('Concentration (nm^-^3)','FontSize',18);
legend('#Vac in MeOx','#Ions in MeOx','Location', 'NorthEast');

plot(vz,n_zhist_V,'-b*',vz,n_zhist_I,'-m','LineWidth',3);

ions_map=[ions_map_lay1; ions_map_lay2];

[n_zhist_I,~] =hist(ions_map(:,3)*1e9,vz);
[n_zhist_V,~] =hist(vac_map0(:,3)*1e9,vz);

normalize=pi*.5*((5)^2);
n_zhist_V=(n_zhist_V./normalize);
n_zhist_I=(n_zhist_I./normalize);

plot(vz,n_zhist_V,'-b*',vz,n_zhist_I,'-m','LineWidth',3);

%-----------------

subplot(3,3,7,'Position',[0.0700    0.05    0.20    0.23]);
%get(gca,'position')
%-----------------
if n<=length(F)
    fid=fopen(['FOutput/Tmap_n',num2str(n)],'rb');
    Tmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz);
    fclose(fid);
end
plot3Dmap(xspacing,yspacing,zspacing(1,11:51),squeeze(Tmap),-10e-9,10e-9);
%Plot_layers(((X2_D-X1_D))),(((Y2_D-Y1_D))),(((layers(1).t))),(((layers(2).t))),'','','','','','',[])
hhb=title('Temperature (K)');
set(hhb,'Color','b');
%p=get(hh,'Position')
set(hhb,'Position',[-8.3552,-9.8902,10.9]);
set(hhb,'FontSize',25);
caxis([300 1400]);
xlabel('x(nm)'],'FontSize',14);
ylabel('y(nm)'],'FontSize',14);
zlabel('z(nm)'],'FontSize',14);
%-----------------
subplot(3,3,8,'Position',[0.3708 0.05 0.20 0.23]);
%get(gca,'position')
%-----------------
if n<=length(F)
    fid=fopen(['FOutput/Vmap_n',num2str(n)],'rb');
    Vmap=reshape(fread(fid,Nx*Ny*Nz,'double'),Nx,Ny,Nz);
    fclose(fid);
end
[~,~,Ezmesh]=gradient(Vmap,xspacing(2)-xspacing(1),yspacing(2)-yspacing(1),zspacing(2)-zspacing(1));
plot3Dmap(xspacing,yspacing,zspacing(1,11:51),abs(Ezmesh),'E-Field (V/m)',-10e-9,10e-9);
hhsd=title('E-Field (V/m)');
set(hhsd,'Color','b');
%p=get(hh,'Position')
set(hhsd,'Position',[-4.1776,-4.9451,10.1]);
set(hhsd,'FontSize',25);
caxis([0 1e9]);
xlabel('x(nm)'],'FontSize',14);
ylabel('y(nm)'],'FontSize',14);
zlabel('z(nm)'],'FontSize',14);

%-----------------
subplot(3,3,9,'Position',[0.6700 0.07 0.16 0.23]);
%get(gca,'position')
%-----------------
plot(n,NMO(n),'LineStyle','none','Marker','o','MarkerFaceColor','red','MarkerSize',11,'Color','black');
hold on;
plot(n,NTE(n),'LineStyle','none','Marker','o','MarkerFaceColor','green','MarkerSize',11,'Color','black');
hold on;
plot(n,NR(n),'LineStyle','none','Marker','o','MarkerFaceColor','black','MarkerSize',11,'Color','black');
hold on;

plot(n,NG(n), 'LineStyle', 'none', 'Marker', '*', 'MarkerFaceColor', 'cyan', 'MarkerSize', 11, 'Color', 'black');

hold on;

% plot(NSp(:,), 'LineWidth', 4, 'Color', 'blue');
% hold on

plot(NMO(:,), 'LineWidth', 4, 'Color', 'red');
hold on

plot(NTE(:,), 'LineWidth', 4, 'Color', 'green');
hold on

plot(NR(:,), 'LineWidth', 4, 'Color', 'black');
hold on

plot(NG(:,), 'LineWidth', 4, 'Color', 'cyan');
hold on

% keyboard;

legend(('#ions MeOx', '#ions OEL', '#Recomb', '#Gen', 'Location', [ 0.835 0.175 0.1201 0.04]));

xlabel('n (# of program steps)', 'FontSize', 14);
ylabel('Events (N.A)', 'FontSize', 14);

set(gcf, 'PaperUnits', 'inches', 'PaperPosition', [0 0 15 9])

opengl('software');
drawnow;

writeVideo(writerObj, getframe(gcf));

print(gcf, '-dpng', ['Transient/maps_n', num2str(n), '.png']);
nn=nn+1;
end
close(writerObj);
end
References


47. Iglesias, V., et al. Dielectric breakdown in polycrystalline hafnium oxide gate dielectrics investigated by conductive atomic force microscopy: AVS.


