A SECURE HFO2 BASED CHARGE TRAP EEPROM WITH LIFETIME AND DATA RETENTION TIME MODELING

By

CHENG HAO

Bachelor of Science in Electrical Engineering
North Carolina State University
Raleigh, NC
2011

Master of Science in Electrical Engineering
Oklahoma State University
Stillwater, OK
2016

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
July, 2019

A SECURE HFO2 BASED CHARGE TRAP EEPROM WITH LIFETIME AND DATA RETENTION TIME MODELING

Dissertation Approved:

Dissertation Adviser Dr. Chris Hutchens

Dissertation Adviser

Committee Member Dr. Jerzy Krasinski

Committee Member Dr. Daqing Piao

Outside Committee Member Dr. Jindal Shah

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisor Dr. Chris Hutchens for the guidance throughout my Ph.D. study. I certainly would not have this achievement without his effort. I also would like to thank Dr. Jerzy Krasinski, Dr. Daqing Piao and Dr. Jindal Shah to be my committee members. They have provided me valuable guidance and suggestions for my work. I would like to thank Air Force Research Laboratory (AFRL), Rome, NY for the funding of this project.

I would like to thank my parents, my mother Mrs. Ru Jia, my father Mr. Genli Hao and other family members for being supportive toward my education. I would like to thank my girlfriend Ms. Hao Guo for being on my side throughout these years. I would like to dedicate this dissertation to all people that support me these years for this great achievement.

I would like to thank my MSVLSI lab members Mr. Vishal Reddy Banala and Mr. Juan Salinas for their help and contributions throughout my study.

Name: CHENG HAO

Date of Degree: JULY, 2019

Title of Study: A SECURE HFO2 BASED CHARGE TRAP EEPROM WITH LIFETIME AND DATA RETENTION TIME MODELING

Major Field: ELECTRICAL ENGINEERING

Abstract: Trusted computing is currently the most promising security strategy for cyber physical systems. Trusted computing platform relies on securely stored encryption keys in the on-board memory. However, research and actual cases have shown the vulnerability of the on-board memory to physical cryptographic attacks. This work proposed an embedded secure EEPROM architecture employing charge trap transistor to improve the security of storage means in the trusted computing platform. The charge trap transistor is CMOS compatible with high dielectric constant material as gate oxide which can trap carriers. The process compatibility allows the secure information containing memory to be embedded with the CPU. This eliminates the eavesdropping and optical observation. This effort presents the secure EEPROM cell, its high voltage programming control structure and an interface architecture for command and data communication between the EEPROM and CPU. The interface architecture is an ASIC based design that exclusively for the secure EEPROM. The on-board programming capability enables adjustment of programming voltages and accommodates EEPROM threshold variation due to PVT to optimize lifetime. In addition to the functional circuitry, this work presents the first model of lifetime and data retention time tradeoff for this new type of EEPROM. This model builds the bridge between desired data retention time and lifetime while producing the corresponding programming time and voltage.

TABLE OF CONTENTS

Chapter Page

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

Trusted computing is currently the most promising security strategy for cyber physical systems which demand high level protection on sensitive data, code, and configurations. Trusted computing has been widely applied to business, e-commerce, medical, industrial, financial entities, mobile computing and government agencies. The trust is in the sense that the relying entity is assured any data given to the system is kept confidential or no malware is running on the platform [1]. Security is bootstrapped from a dedicated secure microcontroller referred to as a Trusted Platform Module (TPM) which is built into more complex computer systems. Today, TPM is prevalently integrated into servers, desktops and laptops for hardware secure authentication. The TPM hardware along with supporting software provides the platform's root of trust. Hardware protected storage enables the protection of user's secret data through the means of encryption whose decryption can only be performed on a dedicated hardware containing the private keys. However, as the demand for TPM involved application increases, the security and trust issues have to be addressed in order not to defeat the original purpose. Research and actual cases studies have shown TPM is subject to serious security threats. The security threats appear both in software and hardware. Our work focuses on providing a feasible hardware solution to protect the TPM structure from hardware attack specifically EEPROM memory attack. We propose an embedded 10kb secure EEPROM architecture employing charge trapping mechanism implemented using 32nm HfO_2 based SOI CMOS process. The architecture includes EEPROM

memory core, fully custom designed communication interface between CPU and EEPROM core, and programming voltage generator. In addition to the hardware implementation, this work provides the first model for EEPROM lifetime and data retention time tradeoff for high k dielectric (HfO_2) type EEPROM. The HfO_2 based dielectric material reliability modeling is still under research. Many research groups have proposed different school of thoughts. This work first summarizes models encountered during initial literature review and then selects a best-fit model among all based on our understanding and feasibility of applying to our EEPROM lifetime model development. Nonetheless, our first model will be available and improvement can be made as deeper understanding of the material physics as the knowledge base grows.

This dissertation is organized as follows: Chapter 2 discusses TPM technology basics, existing memory attacks that potentially threat TPM data security, proposed solutions in the literature, and the perspective of our work to the issue. Chapter 3 discusses about the basics of HfO_2 based high dielectric (HK) constant gate stack, carrier transport and trapping mechanisms, and the potential application for EEPROM. Chapter 4 provides our proposed EEPROM cell structure, CPU/EEPROM communication interface architecture, and operation. Chapter 5 discusses programming voltage generation circuitry. Chapter 6 is dedicated to the proposed EEPROM lifetime modeling. This chapter fist reviews existing SiO_2 dielectric device lifetime models followed by HfO_2 based HK transistor lifetime models. Finally the new model is presented. Chapter 7 shows simulation results of major circuit blocks. Chapter 8 suggests future work.

CHAPTER II

TRUSTED PLATFORM MODULE SECURITY ISSUES

Trusted Platform Module (TPM) is designed to provide hardware-based security functions. TPM is capable of performing number of cryptographic operations. However, TPM is a passive device meaning it cannot enforce any specified software security polices without an operating platform. For example, on a trusted desktop platform, the TPM is assisted by the complex software running from BIOS. TPMs have revealed vulnerabilities can be exploited by skilled malicious users. In fact research and actual test cases have shown TPM is subject to serious security threats. The security threats appear both in software and hardware aspects. Software attack may include Bootloader, BIOS, and system attack. Hardware or physical attack includes memory, bus probing and side-channel interrogation. Secure means such as encryption of on-chip or embedded memory, data bus and address bus encryption, sensing meshes for triggering self-destruction and distributed bus lines over a sea of gates are employed for preventing cryptographic attacks. Usually embedded firmware and encryption keys are stored in nonvolatile memory. Therefore, embedded memory security plays a critical role in trusted computing. Our work focuses on memory security from physical attacks. Our study aims at providing a circuit level feasible solution to direct probing, power analysis and optical observation. This chapter reviews TPM technology and introduces different means of memory physical attacks and proposed security solutions in the literature. Finally a design perspective of our implementation employing hardware at the transistor level to improve storage security is presented.

2.1 Trusted Platform Module

As defined by the Trusted Computing Group (TCG) [2], a TPM is a cryptographic microcontroller designed to secure the hardware authentication process and counteract software attack through integrated cryptographic keys. Safe computing is achieved through authentication and validation. Authentication is the process of proving a platform is what it claims to be and validation helps to prove a platform is trustworthy and has not been breached. TPM provides the platform root of trust and extends the trust to the other parts of the platform by building a chain of trust [1] [3]. The minimum features a trusted platform must employ are protected capabilities, integrity measurement and integrity reporting. In addition to the basic features, TPM has other features allows flexible implementation including confidentiality and integrity protection, secure storage, and process isolation [1]. The following briefly reviews each previously mentioned feature. The protected capabilities of TPM are reflected in specific crypto functions executed within the hardware. External hardware and software are prohibited from direct access to those functions but communicate with I/O ports of the TPM. Integrity measurement is the fundamental feature of a platform as being trusted. This feature tells how the platform is configured and what processes are running. TPM's integrity measurement results in an integrity metric which is compared with acceptable values for verifying the trusted platform. The metric consists of two classes of data; one is measured value which is a representation of embedded data or program code, the other is digests which are Sha-1 cryptographic hash of those measured value [5]. After an integrity metric is obtained, TPM report is made to requesters who make trust decision regarding the platform. The integrity metric must be stored in a secure location since it contains the proof of a trustworthy platform. Confidentiality and integrity protection features allow data and application code to be protected during both storage and execution by encryption. Confidentiality requires stored data to be encrypted. Access to the data requires securely stored cryptographic keys. Process isolation protects data during execution. An isolation kernel between

hardware and operating system is provided to create and manage multiple secure compartments which are in parallel on the same machine. Each compartment runs its own operating system in parallel to and decoupled from others.

A TPM chip architecture is shown in figure 2.1 [4]. The I/O block manages information flow over the communication bus. The I/O block provides encoding and decoding for communication and routes messages to appropriate components. TCG does not specify the I/O architecture and leaves it up to the developers. Nonvolatile memory mainly stores endorsement keys (EK), storage root keys (SRK) and owner authorization data. EK consists of a key pair with private component that is embedded in the TPM and public component contained in the endorsement credential [1]. EK is used for TPM identification and platform extension. To protect user privacy, EK usage is

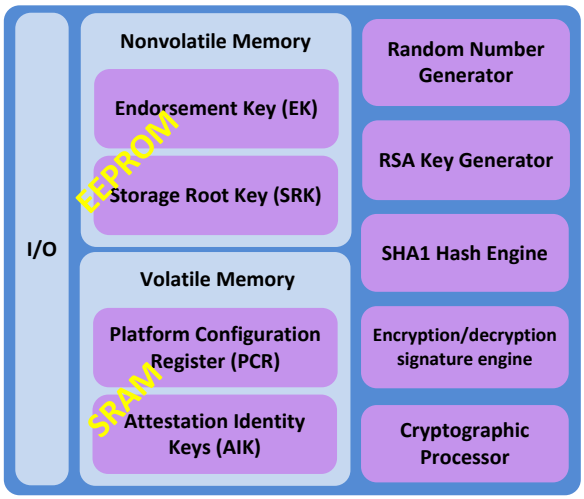


Figure 2.1 Trusted Platform Module chip architecture [4].

restricted and its aliases, the Attestation Identity Keys (AIK), are used for transactions [1]. EK pair is provided by TPM manufactures and stored in tamper resistant nonvolatile memory (enhancing this feature as well as others are the goal of this work). An endorsement credential is generated containing the public EK and security properties of the TPM. A certification authority known as Trusted Platform Module Entity (TPME) can attest and sign the credential after

verifying the public EK whose corresponding private EK stored in the TPM complies with TCG standards [1]. The TPME could be third party users or manufactures. The endorsement credential is used to identify and verify the genuine of the TPM based on stored private EK. The public EK is only used for data encryption during ownership assignment and creating AIK certificates. Public EK encryption ensures data can only be recovered by the particular TPM identified in the endorsement credential. As mentioned, AIK can be seen as an alias of EK. Each TPM supports many AIKs for the TPM user to provide them to different service providers who need to verify the platform identity. Platform Configuration Registers (PCR) are used to store integrity metrics. These registers will be reset when a system loses power or restart. A random number generator is used to construct keys. The SHA-1 engine executes secure hash algorithm used in many cryptographic procedures such as integrity measurements and when computing digital signatures.

The remaining space of this sub-section is used to present an overview of the TPM services: root of trust, boot process, integrity measurement and reporting, protected storage, and attestation. The foundation of TPM services is the concept of "root of trust". The idea is that a trusted platform has to have some means of verifying the integrity of the platform. At the same time, the integrity of the means also need to be verified. Ultimately, there will be some processes that cannot be self verified [1] [2]. This process is the Root of Trust [1] [2]. This is the starting point of the chain of trust among parts in the platform. There are three distinct roots of trust defined by TCG standard: a Root of Trust for Measurement (RTM), a Root of Trust for Storage (RTS) and a Root of Trust for Reporting (RTR) [1]. RTM is a computing engine that boots early in the boot process to enable integrity measurement of other components that loaded after it. RTS is a trusted component that provides protection to keys and data. RTR is a trusted component performing verification to the platform. System booting starts from CPU execute core root of trust for measurement (CRTM), which is the first set of instructions executed for a new chain of trust [4]. The integrity measurement performed together with the boot process and CRTM sends an

identification value to RTS. CRTM serves as a static root of trust for system booting [5]. After BIOS is loaded, it takes control and verifies the integrity of OS loader. The loading and measurement alternatively propagate through the components until applications are loaded. This procedure is depicted in figure 2.2, is adapted from [5]. The integrity measurement consists of two classes of data, one is measured value which is a representation of data or program code, the other is measurement digests which is SHA-1 cryptographic hash of the measured value [5]. The measurement digests are stored in PCR. The verification process requires recreation of digests

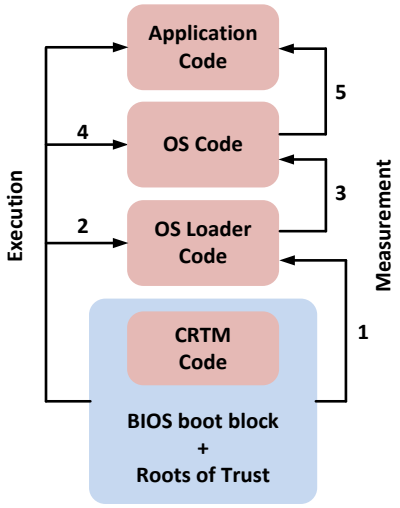


Figure 2.2 TPM system boot process from a root of trust. Source of [5].

and to be compared with PCR stored values. Integrity reporting serves two purposes: open the shielded-locations to store integrity measurements and attests to the authenticity of stored value [5]. Protected storage is one of the critical services provided by TPM. A small amount of volatile memory is used to hold active keys being used for signing and decryption operation. Storage complies with two key attributes: migratable and nonmigratable. Migratable keys may be exchanged between TPM devices following the user. A nonmigratable key is permanently associated with a specific TPM device. Migration of nonmigratable keys will result platform masquerades each other [5], which defeats the secure purpose of TPM and is not allowed to take place. Detailed key specification can be found at [4] and [5]. Attestation is a process by which a

third party or verifier verifies a users' operating system and applications are intact and trustworthy. In order to gain trust from the third party, attestation data should be signed by a TPM whose key is certified by the certification authority [2].

2.2 Means of Memory Attack

Embedded systems often involve microcontrollers which have integrated memory devices such as SRAM, ROM, EEPROM and Flash. System-on-chip (SoC) devices, on the other hand, are more complex compare to microcontrollers and may have multiple processors and/or memory devices integrated on a single chip. Embedded firmware, instruction code and secure keys are often the content of these memory devices. As discussed in 2.1 nonvolatile memory in the TPM stores critical keys for secure device authentication. The ability of sustaining cryptographic attacks determines the usefulness of a TPM device. On-chip memory has less encryption compared to off-chip memory for the following reasons. One, complicated encryption requires multiple instruction fetches which increase latency of program execution. The other reason is strong encryption requires additional hardware which consumes more silicon area [6]. However, the weaker encryption of a memory is more susceptible to physical attacks. Physical attack on encrypted memory can be macroscopically classified into three types: microprobing attack, cold-boot attack, and side-channel attack. This section reviews the physics of each of the four types and provides literature review demonstrates the secure effect have spread to trusted computing applications.

2.2.1 Microprobing Attacks

Microprobing attacks have been known since their application on smartcard processors [7]. It is an attack that directly accesses chip circuitry to observe, manipulate, and interfere with the integrated circuit. The general steps are de-packaging the circuit, reconstruct layout, and manual microprobing or memory content read out [7]. Equipment for carrying out microprobing attack is

relative inexpensive and widely available. The most important tool is a microprobing station which includes a microscope, stage, device test socket, micromanipulators and probe tips [6]. The microscope ought to have long working distance objectives and enough depth of focus to be able to capture all the probes and movements [6]. Figure 2.3 [6] provide a glance of generic scenario of landing probes on a test chip (left) and a microspore image with probes touching the circuit.

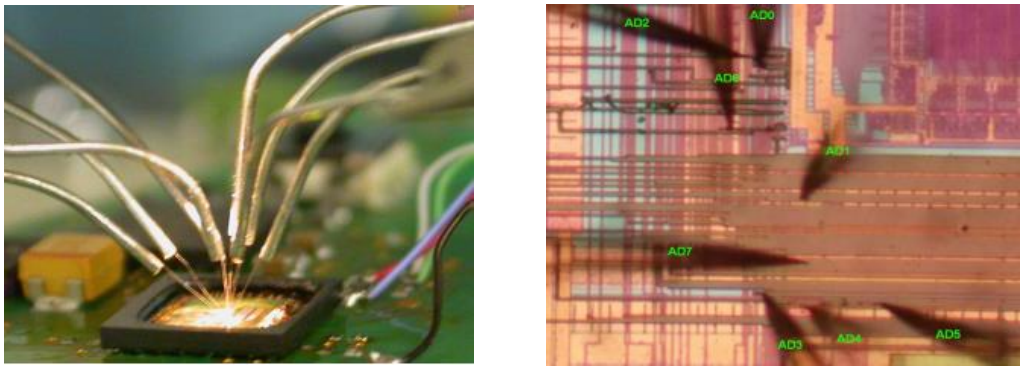


Figure 2.3 (Left) Probing needles land on the chip which is observed by a microscope. (Right) Chip circuit view under a microscope. Marked shadows are probing needle tips. Source of [6].

S. Skorobogatov [6] has demonstrated a successful direct extraction from encrypted embedded memory in secure chips. The Mask ROMs from two samples are the target attack devices. One is a smartcard used in banking industry with likely a Hitachi H8/300 compatible CPU core [6]. The other sample is a custom secure microcontroller used in the car industry with likely a NEC 78K/0 compatible CPU core [6]. After de-capsulation, both chips are found to be fabricated with 0.35um CMOS process. The smartcard chip has 40kB of Mask ROM and 4kB of EEPROM while the microcontroller has 32kB of Mask ROM and 256 bytes of EEPROM. Even though both cores have reduced gaps between metal layer lines and dummy wires, using the technique discussed in [8] authors were able to de-process the chips down to transistor layer revealing Mask ROMs structure. Data was injected into the bus coming from the Mask ROM to determine the correspondence between the encrypted and plain text data. This work raises the question whether memory encryption is as good as it is claimed to be. Perhaps a new way of implementing

embedded memory is in order? K. Kursawe et al. have shown their probing attack to the IBM ThinkCentre M50 TPM [9]. The Atmel TPM is installed on a daughter board. The communication interface is easy to probe. The attack was conducted in two phases. The first phase was using a modified version of the TPM driver to understand the transfer protocols and meaning of signals on the communication bus. The second phase was to observe traffic during the TPM startup. A filter device was added in front of logic analyzer to filter signals that only TPM operation related [9].

2.2.2 Cold-boot Attack

In addition to the microprobing attack, cold-boot attack targets DRAMs which are usually used to store cryptographic key material in computers. This type of attack does not require physical access of the internal circuitry. It uses the phenomenon called memory remanence. Memory remanence means charges stored in DRAM cells do not dissipate away immediately after power is cut or removing the memory from mother board at normal operating temperature [10]. Memory contents can persist even longer if the chip is kept at low temperatures. Attackers can extract RAM contents using three methods with increasing resistance to countermeasures. The simplest method is to reboot the machine and launch a custom program kernel that gives the attacker access to the residual memory content. A relatively strong attack is to restore power to the machine and boot a custom program kernel to extract memory content shortly after power is cut. This method prevents any chance that memory content is scrubbed before power off. A stronger attack is transplant the DRAM modules to an attacker prepared PC after power is cut recovering its data. Data is fully preserved from BIOS or any hardware clearing memory on reboot. Halderman et al. have demonstrated that even after power is cut, encryption keys can be recovered by cooling the memory chip prior to cutting power and apply their developed algorithm for error correction [10].

2.2.3 Side Channel Attacks

One of other susceptible secure exploits is side channel attacks. Side channel attacks (SCA) means monitoring hardware external outputs while cryptographic operations are taking place with the aim of compromising the protected keys or data of a cryptographic device. Rather than decapsulated the core IC, SCA seeks to deduce the secret content from the leakage of information. The assumption behind SCA is outputs of the device, i.e. power supply pins, show correlation with the internal state of the device when executing cryptographic operations [11]. This is particularly true when data is transmitted on serial buses. Common external outputs include heat, execution time, electromagnetic radiations and power consumption [11]. This dissertation reviews exclusively focuses the power analysis attack. Other components of SCA can be found in [12, 13, 14, 15, 16]. Many research groups have demonstrated the success of power analysis attack on cryptographic devices [11, 17, 18, 19]. Power analysis attacks are carried out by measuring the power consumption at the power pins of cryptographic chip. Power analysis attack can be classified into three categories: simple power analysis (SPA), differential power analysis (DPA) and correlation power analysis (CPA). SPA directly interprets power consumption measurements for secure content recovery. It focuses on single power trace or pairs of power trance comparison [18]. With advanced encryption techniques, it is hard to deduce the key by SPA alone, but SPA can predict operation type and algorithm both of which are necessary for key extraction [17]. DPA is more powerful compared to SPA. DPA uses statistical technique to identify difference in power traces even though the difference is buried in noise. The basic method is to partition a set of traces into two subsets and compute the difference of the average of the two subsets. If the choice of which trace is assigned to each subset is uncorrelated, the difference of the subsets will be averaged to zero. Otherwise, the difference will be nonzero [18]. Since the noise is white, even extremely tiny correlations can be isolated with enough trances samples. P. Kocher et al. in their publication [18] reported SPA and DPA on AES-128 encrypted

smart card. The DPA successfully detect power differences corresponding to LSB values of first S-box in round one. Figure 2.4 shows the power trace (from top to bottom) of average of the trace when LSB is 1, LSB is 0, difference of the power and 15 times zoomed difference. The SPA revealed power trace of the smart card triple DES operation shown by figure 2.5. In addition to SPA and DPA, CPA captures secret leakage by finding relationship between characteristics of power trace and a hypothesized power model [11]. The idea is that if a power model is accurate, there should be a strong correlation between predicted output and actual output. One power model can be used is

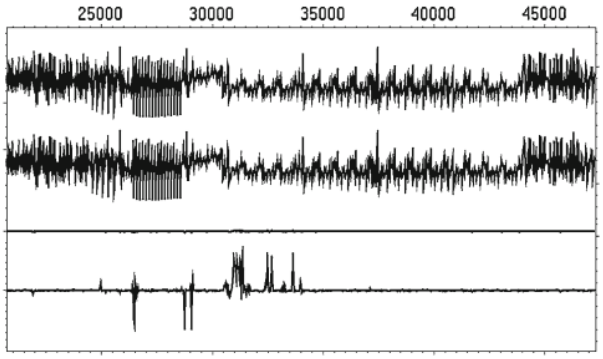


Figure 2.4 The components of a typical successful DPA result. From top to bottom is the average of the trances where the LSB of the output of first S-box in round one is 1, the average of trances where the LSB is 0, the difference between the top two traces, and the same difference with Y axis magnified 15 times. Horizontal axis is time. Source of [18].

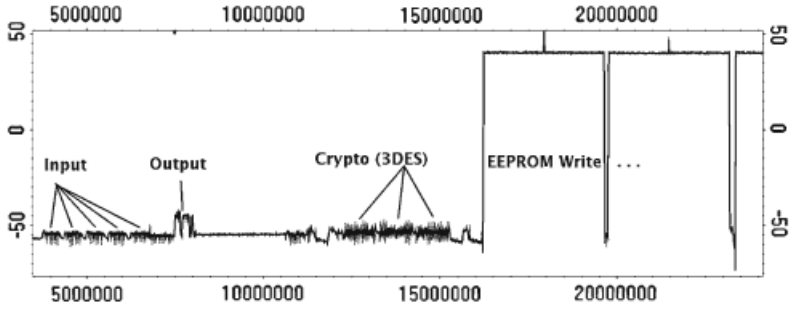


Figure 2.5 Power trace of smart card performing triple DES based operation. Source of [18].

the Hamming Weight Power Model. The assumption of using this model is that the number of 1 or 0 of an output correlates to the power consumption [11]. The correlation between power model

and actual power consumption can be calculated using Pearson correlation coefficient equation [11]. O. Lo et al. [11] have demonstrated the feasibility of CPA against the AES-128 algorithm on Arduino Uno microcontroller by monitoring the power consumption while running cryptographic operations.

The success of such attacks are based on the observation that pad drive power is more than three orders magnitude larger than on chip bus power and readily observed when data is transported from off chip onto a serial bus. The observation becomes much less observable when EEPROMs are moved on chip with data or keys transported in parallel for the following reasons 1) Bus power consumption drops by three orders of magnitude or greater when on chip and 2) signal to noise ratios drop by another two orders of magnitude when transport via a 128 bit parallel bus. This may not provide total key or data security from recovery however it does make recovery via power supply monitoring more than five orders of magnitude more difficult excluding the presence of multicore activity.

2.3 Variety of Hardware Security Means

There are not many countermeasures to physical attacks on the memory of TPM have been published. Among limited published literatures we select two representative and interesting countermeasures for discussion in this document. P. A. H. Peterson proposed a software-encrypted virtual memory manager, cryptkeeper, to mitigate the vulnerability of RAMs [20]. The idea is to divide a RAM into two parts, one smaller and unsecure segment for immediate memory use, the other is larger and a secure segment. Even though the paper does not give any information on cryptkeeper implementation and how it can improve RAM security but only the performance assessment, it hand waves the concept of secure RAM by swapping free and occupied memory space. P. Choi and D. K. Kim proposed a new TPM architecture based on a physical unclonable function (PUF) to protect stored secret keys from physical attacks. The idea

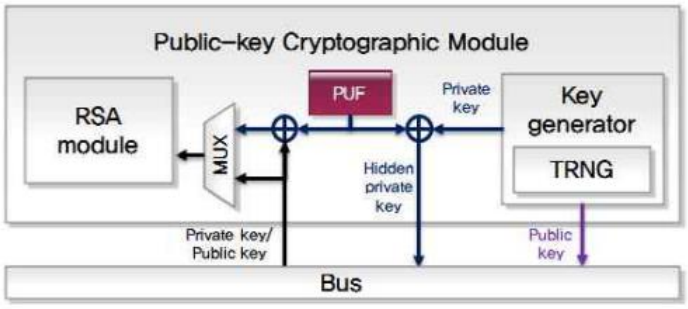


Figure 2.6 Proposed RSA module by P. Choi and D. K. Kim. Source of [21].

is to hide the key before it leaves key generator and unhide it when used in cryptographic modules [21]. Figure 2.6 shows the implementation of PUF with the RAS generator to generate keys. The public key generated directly goes to data bus. The private key is hidden by XORing the key with PUF. The PUF is a function based on the random mismatching of electric elements that is impossible to duplicate [21].

2.4 Perspective of This Work

Beyond the architectural level secure solutions introduced in the literature, we turn our attention to the fundamental building element, the transistor. The inspiration comes from hiding the key (or logic 1 and 0) in terms of transistor threshold difference. Threshold difference between two transistors is used to distinguish logic high and low. The difference of threshold is achieved by programming transistors. The programming is in the sense of applying higher gate to source voltage so that the carrier can tunnel through the oxide barrier and trap into the gate dielectric. This kind of transistor, known as charge trapped transistor (CTT) [22], uses high dielectric constant (HK) material as gate oxide. F. Khan et al. [22] and C. Kothandaraman et al. [23] have demonstrated the feasibility of this kind of transistor for potential memory applications. As a potential solution to the above discussed memory security issues, we propose an on-chip 10-kb EEPROM architecture employing CTT in this work. One interesting aspect to note is that this HK transistor is the product of continuous CMOS technology scaling. As transistor dimensions

become smaller and smaller, the gate dielectric (SiO_2) thickness reduces to the point where gate leakage and reliability are not acceptable. Therefore, a HK material is introduced to the technology for smaller equivalent oxide thickness (EOT) but thicker physically to alleviate the burden of aggressive scaling. However, HK materials have the property of trapping charges in the dielectric after extended time exposure to normal gate voltage applications or short duration higher gate voltage. Many researchers including ourselves see this as an opportunity for a threshold controllable device and a potential "hidden key" memory use. The design presented here uses 32nm SOI CMOS technology with HfO_2 gate dielectric material. The concept of CTT resembles floating gate technology but requires much lower programming voltages and is CMOS process compatible. Detailed device operation is discussed in Chapter 3. The proposed EEPROM uses transistor charge trap property and differential cell structure to distinguish logic one and zero. There is no additional hardware required for security purpose. The process compatibility allows the key containing memory to be embedded with the CPU core. This means the TPM and hosting CPU can be integrated into one chip. This all but eliminates the eavesdropping on the communication buses and power busses. TPM and hosting CPU can be integrated using ASIC design methodology which results in a sea of gates and distributed interconnected wires. This makes locating the storage hardware and reverse engineering the logic a very difficult or an extremely high cost and difficult task. The power analysis attack is prevented by our design as the data transferred in parallel between CPU and EEPROM on chip while consuming three or four orders of magnitude less power or typically 80μ watts in the presents of 100 watts of processor power. Malicious users may only be able to measure the average power consumption of the cryptographic operations considering they are attempting to measure microwatts of power. It is expected to be difficult to observe power fluctuations when cryptographic operations are taking place due to both the parallel nature and lower power consumption levels relative to the processor(s). Finally, even if the memory is found, key need only to be updated when a TPM falls into the wrong hands.

CHAPTER III

HFO_2 HK DIELECTRIC TRANSISTOR

The fundamental building transistor of our work is the charge trap transistor based on HfO_2 HK dielectric gate stack structure. A charge trap transistor is able to trap and detrap carriers into and out of the HK layer under programming or erase gate voltages. The EEPROM cell presented here uses NMOS for charge storage cell. Therefore, the programming voltage is a positive gate voltage with respect to source terminal and erase gate voltage is negative. This chapter is dedicated to a discussion of HfO_2 HK transistor. Topics include HK gate stack structure, defect levels in the HK material, carrier transport mechanisms in the HK gate stack, device threshold instability, memory application, and reliability issues of the gate stack.

3.1 High-k Dielectric Gate Stack

3.1.1 Why High-k

For decades, the semiconductor industry has made its impact through transistor scaling by shrinking transistor gate length, gate oxide thickness, and width, improving performance through lower power, greater speed, and high packing densities. However, continuous scaling will eventually reach or may have reached its limit where gate leakage and gate oxide reliability are unacceptable. After thinning gate oxide (SiO_2) below 2nm, gate leakage current due to direct tunneling phenomenon of carriers through the silica becomes too high to continue [24]. Therefore, at the 65nm node and below new gate materials were introduced to continue the scaling [25]. The solution has replaced SiO_2 with higher dielectric constant materials. The

concept is summarized in equations (3.1) and (3.2)

$$\frac{\varepsilon_{SiO_2}}{t_{SiO_2}} = C_{ox} = \frac{\varepsilon_{HK}}{t_{HK}} \tag{3.1}$$

where ε is material dielectric constant and t is thickness. For the same equivalent gate capacitance in accumulation region, the higher dielectric constant material will be physically thicker to reduce or eliminate the tunneling effects. Silica thickness becomes the equivalent oxide thickness (EOT) which the new material would provide, as shown by equation (3.2),

$$t_{SiO_2} = EOT = \frac{\varepsilon_{SiO_2}}{\varepsilon_{HK} t_{HK}} \tag{3.2}$$

3.1.2 Choices of Material

High k materials generally are defined as those having dielectric constants higher than 9 and include metals from group III-V, lanthanides and aluminum [25]. Table 3.1 summarizes the potential silica substitution HK materials that have been studied in the past decade [26] [27]. HK materials are required to meet gate leakage requirement for low power application. For the first generation HK gate material, integration has shifted to HfO_2 and Al_2O_3 which leading to an approximate EOT of 17Å [28] [29]. Continued scaling below EOT of 10Å requires even higher k material such as La_2O_3. Criteria for selecting HK oxide materials include [24] (1) enabling

Table 3.1 Material being pursued as a potential replacement of SiO_2.

Material	k	E_g (eV)	CBO(eV)	VBO(eV)
Si_3N_4	7	5.3	2.4	1.8
Al_2O_3	9	8.8	2.8	4.9
La_2O_3	30	6	2.3	2.6
ZrO_2	25	5.8	1.5	3.2
HfO_2	25	5.8	1.4	3.3
$HfSiO_4$	11	6.5	1.8	3.6

continuous EOT scaling for a reasonable number of technology nodes, (2) the material has to be thermodynamically stable with Si, (3) it has to have enough band offset with Si (~ 1eV) to

minimize carrier injection [30], (4) minimize threshold instability due to high defect density, (5) limit mobility degradation, (6) sustain gate reliability, (7) must form good electrical interface with Si. While seeking a high dielectric constant material is the objective, very large dielectric values result in an unwanted fringing field from the gate to the source and drain. This fringing field reduces gate control ability and degrades short channel performances [31].

3.1.3 Gate Stack Structure

Figure 3.1 shows the cross-section view of HK dielectric gate stack structure. It consists of a SiO_2 interfacial layer (IL), a deposited HK layer, a metal gate layer, and a poly-Si contact layer. Each layer serves a specific purpose. The poly-Si layer controls the gate height in the integration

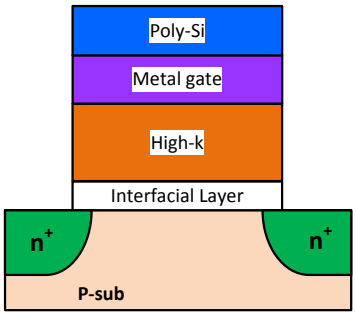


Figure 3.1 High-k dielectric metal gate stack.

process and prevents excessive oxygen exposure to metal gate and HK layer as well as minimizing IL thickening and oxidation of the metal gate [32] [33]. The metal gate eliminates the poly-Si depletion effect and provides work function control [32] [33]. The HK layer is a deposited oxide providing the EOT requirement [32] [33]. In our application the HK layer serves the purpose of storing trapped charge for memory operation. The IL (SiO_2) is formed by oxidation during deposition process combined with oxygen diffusion during the subsequent heat cycle [34]. There are two metal gate/HK transistor integration schemes, gate first and gate last [26] [35]. Gate first integration involves deposition of HK dielectric and metal gate followed by poly-Si gate, patterning gate lines, source/drain implants. A dopant activation anneal at 1000°C is

carried out. In gate last or replacement gate integration, poly-Si-capped dummy gate stacks is deposited first. These stacks are patterned for defining gate and source/drain regions. After implants and the activation anneal, gate materials are replaced with final stack. Figure 3.2 shows the process steps for gate first and gate last respectively [36].

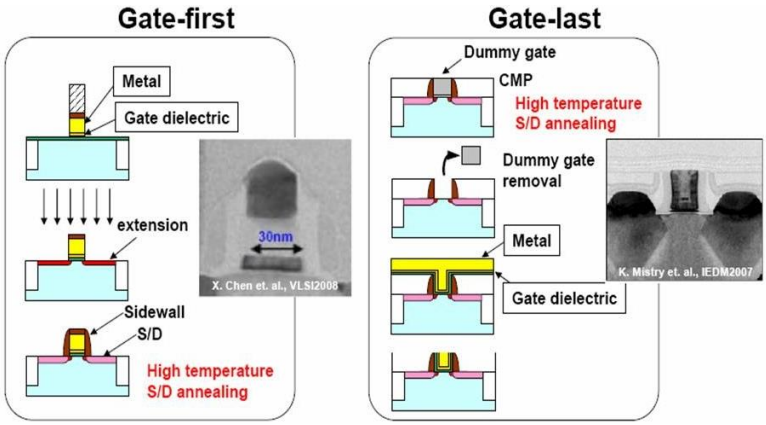


Figure 3.2 Gate first and gate last process steps. Source of [36]

3.2 Defect Levels

Gate dielectrics are required to have as few as possible electrically active oxide defects. Electrically active defects are defined as atomic configurations that located within the dielectric material bandgap capable of trapping carriers [24]. Unlike silica, HK oxide materials have ionic bonding structure and a high coordination number. Thus, intrinsic defects such as oxygen vacancies, oxygen interstitials, or oxygen deficiency due to valency of the metal exist in the lattice [24]. Among these types of defect, oxygen vacancy and oxygen interstitial have the lowest formation energy in HfO_2 [37]. The defect density levels found by amplitude charge pumping (ACP) correlate linearly with stress voltage that indicating oxygen vacancy is the primary cause of charge trapping in HfO_2 [26]. Xiong et al. have used the screened exact exchange (sX) method and the weighted density approximation (WDA) to derive the energy levels. The band gap of bulk HfO_2 is calculated to be 5.6eV, 5.95eV, and 5.75eV in sX for cubic, tetragonal, and monoclinic phases respectively [38].

Possible electron traps in the bulk of HfO_2 are the oxygen vacancy levels calculated to be V_O^{2+}, V_O^{+}, V_O^{0}, and V_O^{-} [26] [37] [38] [39]. Figure 3.3 [38] shows the energy levels of oxygen vacancies and interstitial in the band gap of HfO_2 with respect to Si. V_O^{2+} is at shallow level which is close the conduction band (CB) of HfO_2 and move to V_O^{+} level after trapping an electron. After trapping another electron, V_O^{+} moves to deeper V_O^{0} level. One should notice that V_O^{0} level is within the Si bandgap and it contributes to the memory application as will be shown in Chapter 6. Oxygen interstitials in HfO_2 bulk are responsible for hole trapping. The I_O^{-} level is a half-filled state at 1eV above the valance band (VB) of HfO_2. The I_O^{0} has two energy state. One is empty σ* state located at 4.2eV in the upper bandgap. The other is a filled π* state just above the VB edge, as shown by figure 3.3 [40]. The I_O^{+} ion also has an empty σ* state located near the CB edge and half-filled π* state close to the VB edge [40].

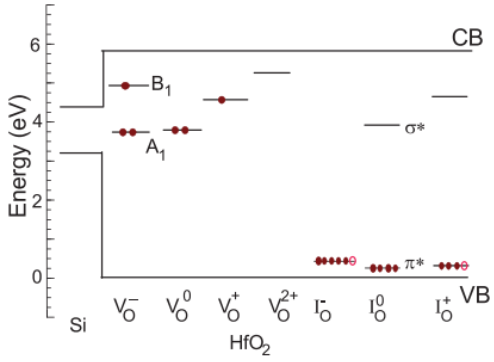


Figure 3.3 Calculated energy levels of oxygen vacancies and oxygen interstitial in HfO_2. Source of [38].

Defect energy levels relative to Si conduction band edge after carrier trapping determines the use as a memory. Filled defect levels within the Si bandgap are preferred for memory applications. Charge trapped in these levels have sufficient retention time and can be removed by a negative gate bias voltage. Filled defect levels above the conduction band edge of Si contributes to fast charge transfer. Carrier stored in these levels can escape easily after programming voltages are removed. Filled defect levels below the valance band of Si give rise to the fixed charge which

cannot be removed easily with ordinary erase voltages. Thus, the defect energy levels of the material determines the feasibility of memory application.

3.3 Carrier Transport

Carrier transport mechanisms in HK gate stack can be classified into two categories: intrinsic and extrinsic mechanisms. The intrinsic transport mechanisms are considered always present even when defect free dielectric film is assumed. The extrinsic transport mechanisms are considered when relating to the presence of defects [41]. The intrinsic transport mechanism includes direct tunneling, Fowler-Nordheim (FN) tunneling, and Schottky emission (SK). *The extrinsic transport mechanisms include carrier elastic/inelastic tunneling into and out of defects known as trap assisted tunneling (TAT), carrier tunneling between defects known as trap-to-trap tunneling (TT), and Poole-Frenkel (PK) tunneling which is field enhanced thermal emission of electron from defects.* Figure 3.4 shows the HK gate stack energy band diagram under positive gate voltage with all the carrier transport mechanisms.

Direct tunneling refers to carrier tunneling through the full barrier height of the SiO_2 layer determined by the conduction band offset of the materials. Tunneling current density is given by [42]

$$J_{dir} = A \cdot E_{ox}^2 \cdot \left(\frac{\Phi_B}{V_{ox}}\right) \cdot \left(\frac{2\Phi_B}{V_{ox}} - 1\right) \cdot e^{-\frac{B\left(1-\left(1-\frac{V_{ox}}{\Phi_B}\right)^{\frac{3}{2}}\right)}{E_{ox}}} \tag{3.3}$$

where A and B are constant, E_{ox} is the electric field across the dielectric, V_{ox} is the voltage applied to the oxide, Φ_B is barrier potential. At higher applied voltage which results a steeper band bending. Carriers tunnel through the triangular energy band and results Fowler-Nordheim

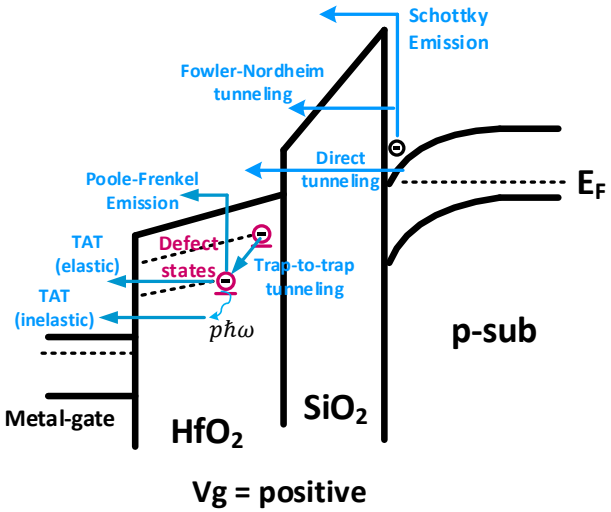


Figure 3.4 Summary of carrier transport mechanisms in HK gate stacks.

tunneling as shown in figure 3.4.

The FN tunneling current density is given by [43]

$$J_{FN} = A_{FN} E_{ox}^2 e^{-\frac{B_{FN}}{E_{ox}}} \tag{3.4}$$

where A_{FN} and B_{FN} are given as

$$A_{FN} = \frac{q^2}{8\pi h \phi_0} \tag{3.5}$$

and

$$B_{FN} = -\frac{4}{3}\sqrt{\frac{8\pi^2 m}{h^2}}\frac{(q\phi_0)^{3/2}}{q} \tag{3.6}$$

As the conduction band offset of the SiO_2 layer is low, emission of carriers over the SiO_2 barrier into the conduction band of the HK film allows for thermionic emission. For high temperatures and low electric fields emission of electrons over the SiO_2 barrier dominates current contribution, while for low temperatures and high electric fields tunneling of electrons dominates. The former case is called Schottky emission and is given by

$$J_{SK} = AT^2 e^{-\frac{1}{K_B}\left(\Phi_B - \sqrt{\frac{q^3 E_{ox}}{4\pi\varepsilon_{ox}}}\right)} \tag{3.7}$$

where A is a constant, T is temperature, K_B is Boltzmann constant, Φ_B is barrier potential of SiO_2, ε_{ox} is oxide permittivity. The latter case is also known as FN tunneling.

Besides direct tunneling and SK emission, which are one-step tunneling processes, defects in the dielectric layers give rise to tunneling processes based on two or more steps. For our application, this tunneling component is observed after write-erase cycles due to the generation of more defects in the dielectric. The increased tunneling current at low gate stress is called SILC and is mainly responsible for the degradation of the retention time of nonvolatile memory devices [44] [45] [46]. The SILC formation is explained by TAT [47] [48]. Reference [49] has summarized the current density for trap-to-trap conduction, equation (3.8) and PF emission, equation (3.9),

$$J_{TT} = C_1 E_{ox} e^{-\frac{E_a}{k_B T}} \tag{3.8}$$

where C_1 is a constant, E_a is activation energy,

$$J_{pf} = E_{ox} \cdot e^{-\frac{q\left(\Phi_B - \sqrt{\frac{q \cdot E_{ox}}{\pi\varepsilon_i}}\right)}{kT}} \tag{3.9}$$

As shown by figure 3.4, write operation is governed by direct tunneling or Fowler-Nordheim tunneling mechanism depending on the applied electric filed strength. During write operation, carriers tunnel through the SiO_2 interfacial layer and trap into the defect states in the HK layer. Defect states shift to lower energy state after capturing carriers as discussed in section 3.2. Depending on the applied electric field strength, the amount of band bending of SiO_2 interfacial layer determines the tunneling mechanism to be direct or Fowler-Nordheim. For high programming electric filed strength, carriers tunnel through the triangular part of the interfacial layer conduction band resulting Fowler-Nordheim tunneling. Trap-to-trap tunneling, TAT, and

Poole-Frenkel tunneling govern carrier transport with in the HK dielectric and relates to the stress induced leakage current stage of breakdown process discussed in Chapter 6. As the dielectric layer is stressed after several programming cycles, new defects are generated with the HK layer. Carrier's mean transporting distance is deduced due to the new defect sites. Thus, gate leakage shows observable increase with the help of Trap-to-trap tunneling and TAT tunneling. As more defects are generated, gate dielectric material starts wearout process. Knowing the carrier tunneling mechanisms enhances EEPROM designer's ability to more accurately predict and model circuit lifetime.

3.4 Threshold Voltage Instability and Memory Applications

3.4.1 Charge Trapping/Detrapping and Threshold Voltage Instability

As discussed in the previous section, HfO_2 HK material possesses pre-existing defects that contribute to carrier trapping and therefore threshold shifts. The trapping and detrapping occurs in both the HK and IL layers. The trapping/detrapping can either be fast or slow depends on the destination defect level. Charge trapping occurs when the gate voltage is ramping up and carriers gain sufficient energy to tunnel through the IL layer reaching a trap site in the HK layer. Charge trapping reveals positive (NMOS) threshold shift on I_g-V_g measurements. Charge detrapping occurs under two conditions. One is during sense measurement where the stress voltage is removed. This kind of detrapping usually comes from the carriers that are trapped in the shallow defect levels close to the IL layer tunnel back to the Si-substrate. The other detrapping is achieved by applying a revers bias to the gate to remove carriers from trap sites. This detrapping is for the carriers that trapped in the deeper energy levels in the HK layer and relative to EEPROM eraser. This means detrapping is not spontaneous but needs the assist of an external reverse bias. Figure 3.5 shows HfO_2 gate stack energy band diagram for carrier trapping and detrapping into and from the HK layer.

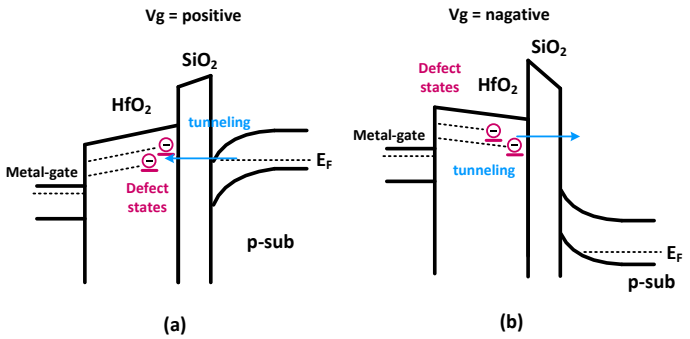


Figure 3.5 Energy band diagram describing charge trapping mechanism for NMOS. (a) Positive gate voltage results carriers tunneling through the SiO_2 layer and trapped into the HK HfO_2 layer. (b) Negative gate voltage results carriers de-trap from the HfO_2 layer.

As electrons trap and accumulate in the HfO_2 dielectric of NMOS, the primary effect is shifting of the threshold voltage. Similarly, hole trapping results threshold shift in the PMOS. Methods of studying threshold voltage instability characteristics utilize both DC and AC stress voltages. Threshold shift due to charge trapping is a function of gate stress, temperature, and stressing time [29]. Table 3.2 shows some selected examples from literature showing threshold shit measurements due to charge trapping.

Table 3.2 Examples of threshold instability characterization

Reference	Gate Stack	Stress Conditions	Threshold Voltage shift
[28]	n^+ poly-Si/HfO_2/SiO_xN_y/p-Si (100) HfO_2 films are deposited by atomic layer deposition (ALD) at 300 °C. Interfacial layer is thermally grown in nitric gas ambient. HfO_2 layer thickness is 3nm. SiO_xN_y IL thickness is less than 1nm.	Vg = 1.5V for ~9000s until ~80mV Vt shift. Followed by Vg = -0.7V at room temperature.	Detrapping results Vt level off at ~50mV suggesting the existence of deep traps where carriers remain.
[50]	n^+ poly-Si/HfO_2/SiON/p-Si SiON IL formed by rapid thermal annealing in NH_3 ambient at 700 °C after Si surface cleaning with Hf solution. HfO_2 is deposited by reactive DC magnetron sputtering method in Ar/O_2 ambient. Followed by post-deposition annealing (PDA) at 500 °C.	DC stress 1.5V to 2.1V during 1000s. Temperature range is 25 – 120 °C. AC unipolar stress with 50% duty cycle. Same voltage and stress time as DC. Frequency range is DC to 1 MHz.	Initial Vt is -0.16V. DC stress results ΔVt = 11mV – 35mV AC unipolar stress exhibits less ΔVt. Higher stress frequency, less ΔVt.
[51]	Si surface cleaning using O_3 based chemistry results a ~1nm SiO_2 IL. HfO_2 is deposited using atomic layer chemical vapor deposition (ALCVD). HfO_2 thicknesses are 3, 4, and 6nm. W=20µm, L=20µm.	Constant voltage stress (CVS) for 1s. Maximum electric filed applied is 5.5 MV/cm.	Threshold shift of ~30 – 50 mV
[29]	n^+ poly-Si/HfO_2/SiO_2/p-Si HfO_2 is deposited by ALD with thickness of 3nm. IL layer is ~1.1nm. L>1um, W= 20 to 50um.	AC unipolar stress pulse. $V_{d,\,sense}$ = 50mV. Vg = 1- 2V. Temperature range is 25 - 180 °C.	Threshold shift as a function of gate stress voltage and temperature. At 25°C, Vt shift = ~10mV at Vg = 1V. Vt shift = ~90mV at Vg = 2V for stressing 1000s. At 180 °C, Vt shift = ~80mV at Vg = 1.5V for stressing 1000s.

3.4.2 Memory Applications

The concept of a charge trap transistor used for nonvolatile memory has been in existence for several decades. The gate stack has the structure of a silicon-oxide-nitride-oxide-silicon (SONOS) [52] [53]. Silicon nitride, tantalum and titanium oxide were commonly used. However, due to the low conduction band offset of tantalum and titanium oxide, they were gradually removed from the list [24]. Silicon nitride charge trapping layers in SONOS memory structure were also investigated extensively with the conclusion they offer poor retention and suffer scaling issue [54]. Therefore, to improve program/erase speed, vertical scaling and charge retention characteristics of nonvolatile memory device, hafnium based oxide turns out to be used for charge trapping layer for newer generation nonvolatile memory device. In the previous sections, we have discussed trap states existing in the HfO_2 band gap and threshold shift due to carrier trapping in those states. Indeed, HfO_2 dielectric multiple-time programmability is possible for nonvolatile memory application. Khan et al. [22] are the first group to have demonstrated the programmability of the high k transistor, or charge trap transistor (CTT). Figure 3.6 depicts the basic operation of CTT. When positive gate voltage is applied, carriers trap into the HK layer

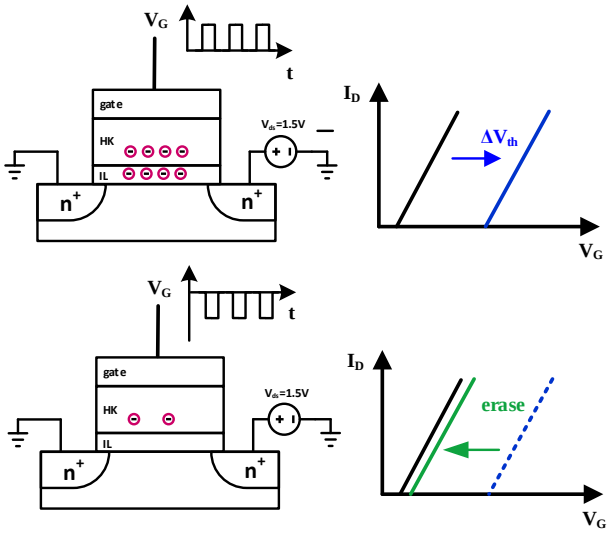


Figure 3.6 Threshold voltage increases when carriers trapped into the HK dielectric (top) and decreases when carriers are detrapped (bottom).

resulting in a positive (NMOS) threshold voltage shift. When a negative gate voltage is applied, carriers detrap from the HK layer and threshold voltage shift back to near its native value. To further investigate the program/erase characteristics, Khan's group stress 1.2um x 20nm (22nm SOI technology) devices with a gate ramp voltage of 10ms pulse of 10mV increments [22]. Higher drain voltage and temperature results in enhanced trapping and more stable threshold shift. This means the effect of programming not only depends on gate stress voltage but also the drain voltage for supplying hot carriers. One should notice that there is an obvious trade-off between trapped charge retention and required erase time/voltage. Higher programming voltage can lead to more stable threshold voltage shift, but requires longer time or higher voltage to erase [22] [55]. Under the programming and erase conditions $V_{g,\ program}$ = 2V, V_d = 1.2V and $V_{g,\ erase}$ = -2V, a memory window of ~120mV is shown even after 800 program/erase cycles [22]. In addition to Khan's group, Kothandaraman et al. have shown that 2V gate stress for 1ms with 1.5V drain voltage results excess of 100mV V_{th} shift [23]. The extrapolated loss in V_{th} is only 16% over 10 years under 85°C [23]. Jayaraman et al. have shown with ~10ms 2V gate pulse and 1.5V drain voltage, V_{th} exhibits ~200mV shift and ~100ms gate voltage pulse for ~300mV V_{th} shift [56]. More than 10 years lifetime of HfO_2 based memory application is also confirmed by [55].

3.5 Reliability Issues

3.5.1 Bias Temperature Instability

Bias temperature instability (BTI) is a phenomenon that transistor threshold voltage shift due to carriers trap into the pre-existing defects under constant voltage stress and elevated temperature [57]. This phenomenon exists for both PMOS and NMOS namely negative BTI (NBTI) for PMOS and positive BTI (PBTI) for NMOS. PBTI is attributed to carrier tunneling through the IL layer and trapped into the HK layer causing threshold voltage increase [58] [59]. NBTI, on the

other hand, is driven by interface states density and influenced by nitrogen in IL [51] [60]. For our application, BTI is not a critical issue. In fact we need to uses the programmable threshold to achieve logic one and zero difference for memory operation. The only major concerns for us is the minimum threshold shift that can be sensed after a desired storage time and lifetime degradation due to the threshold programming. Or more specifically what is the least damaging write protocol within a desired write read cycle for a desired EEPROM cell lifetime. The detail is embedded into the memory modeling in Chapter 6.

3.5.2 Stress Induced Leakage Current

Stress induced leakage current (SILC) generation is a phenomenon that gate current increase under the bias temperature stress. It is attributed to random or local defect generation in the HK layer depending on amorphous or polycrystalline structure of the dielectric and explained by TAT [47] [48] [61]. An example of SILC study [62] is used here to illustrate the phenomenon. Device with gate stack structure TiN/HfO_2/SiO_2/p-Si is stressed by stress-sense procedure as shown in figure 3.7 [62]. Several interesting and important observations have been noticed. First, SILC is temperature dependent. At higher temperature, a linear increase in the SILC with stress time is observed. This linear increase is due to the creation of new defects in the HK layer [62]. The contribution of the increase may consist of three components: direct tunneling, TAT filling pre-existing defects and TAT creating defects. Second, SILC is a reversible process. In this example, after application of reverse bias of range $-2V < V_g < -1V$, SILC fully relaxes. Larger threshold shift and high SILC are observed after stress suggesting that new defects are generated. The damage during the PBTI stress is irreversible [62]. Perhaps the most important consequence of SILC is its contribution to dielectric breakdown. SILC is referred as time dependent pre-breakdown degradation of the dielectric [63].

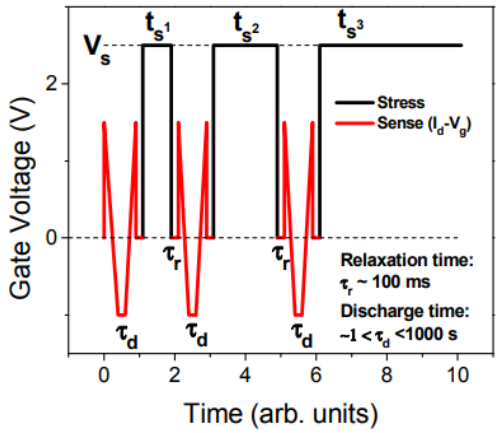


Figure 3.7 Stress-sense procedure for SILC generation measurement. Stress voltage V_s is 2.5V. t_s is stress time. τ_r is stabilization time. τ_d is discharge time. Source of [62].

3.5.3 Time Dependent Dielectric Breakdown

Time dependent dielectric breakdown (TDDB) is a phenomenon that one or more conducting path formed in the dielectric material connecting transistor gate and substrate due to a period of gate voltage stress. The cause is generally understood as defects generated in the bulk of the dielectric leading to gradually formed percolation paths and eventually making the dielectric material lose its insulating property. TDDB of SiO_2 gate dielectric has been studied for several decades. Among many proposed models, there are four frequently used models to describe the TDDB: the E model, the 1/E model, the power law model, and the root-E model. Details of each model are given in Chapter 6. As the HK materials replacing SiO_2 for newer technology nodes, their TDDB characteristics have to be understood. Significant amount of research effort has been devoted to understanding and modeling of HK TDDB mechanism. However, a commonly agreed universal model has yet to be introduced. This section provides an overview of TDDB for a HK dielectric gate stack. Representative TDDB models for HfO_2 HK dielectric from literature and our TDDB model development for our memory application are given in Chapter 6.

The critical part of TDDB modeling is to detect the first occurrence of dielectric breakdown. Furthermore, some MOS digital circuits can still function after gate breakdown given that post

breakdown resistance is high enough [57]. Thus, it is accurate to separate breakdown into soft (SBD), progressive (PBD), and hard breakdown (HBD) [57]. For an amorphous structure, SBD and HBD are randomly distributed over the lattice. For polycrystalline structure, breakdown is localized [61] [64]. There is controversy on occurrence order of the three breakdown components. Ribes et al. [57] suggests that SBD is not a precursor of HBD. Both SBD and HBD coexist. However, Pagano's group [65] and Bersuker's group [66] believe otherwise. Regardless the controversy, progressive breakdown is now the well accepted breakdown mechanism.

The statistics of TDDB obey Weibull distribution [57] [67] given by

$$\ln[-\ln(1-F)] = \beta \cdot \ln(t) - \beta \cdot \ln(\alpha) \tag{3.10}$$

where F is percentage failure, β is Weibull slope or shape parameter, α is characteristic time-to-failure. The term on the left is also known as the Weibit. Zero Weibit corresponds to 63.2% of failure and is usually used to describe the device mean-time-to-failure. Weibull slope, β, represents the failure rate behavior,

- $\beta < 1$, failure rate decreases with time
- $\beta > 1$, failure rate increase with time
- $\beta = 1$, failure rate is constant

Different breakdown segments (soft and hard) have different Weibull slopes [68] [69]. The different Weibull slopes mean breakdown at one site is independent of others sites. It may also suggest different breakdown rates at different sites. Without knowing the detailed physics of breakdown mechanism, lifetime of the HK device can be simulated using kinetic Monte Carlo algorithm. However, in order to use kinetic Monte Carlo method, the defect generation rate(s) have to be known or assumed. Different generation rate equations have been provided by different research groups [70] [71] [72]. For serving the purpose of theoretical model development, we use three dimensional kinetic Monte Carlo method to extrapolate lifetime based

on a desired programming electric field. The electric field strength is derived from required data retention time of the proposed EEPROM. Therefore, an optimal programming electric filed or programming voltage that balances the tradeoff between lifetime and data retention time can be obtained from our model. Chapter 6 discusses model details.

CHAPTER IV

PROPOSED EEPROM ARCHITECTURE

This chapter documents the implementation of the proposed 10-kb EEPROM. The EEPROM core consists of four 256x88 bits memory banks. The CPU and EEPROM interface provides data communication and memory core operation control. The programming voltage generation and control are design to be achieved by on-chip low-dropout voltage regulators (LDO) and off-chip DACs. Programming generation block is detailed in Chapter 5. The chapter layout is the following: design overview, CPU/EEPROM interface, interface operation, the EEPROM core.

4.1 Design Overview

Figure 4.1 shows the top level block diagram of the EEPROM. The CPU/EEPROM interface provides data communication scheduling between CPU and EEPROM core, programming and memory operation control. The interface consists of three input registers, CUI (command-user interface) unit, counters, clock divider, and controller. The on-chip LDOs together with off-chip DACs provide accurate and adjustable programming voltages that allow programming voltage to be applied in a ramp fashion to avoid application of excess voltage and protect memory cells. The proposed write voltage range is from 1.5V to 2.7V across gate to source terminals and the erase voltage is from 0 to -1.5V. Read voltage range can be from 0.5V to 0.9V. Discrete DACs have ±3V output range. In order to use DAC's full bits of accuracy discrete summing stage can be implemented with each DAC to shift the DAC output to the desired voltage range. CPU interfaces the EEPROM with four signals: data, CPURW, address, and clock. Data contains the information

being loaded into the input registers which either be EEPROM operation/setup commands or actual memory data. CPURW is a flag signal indicating CPU read from or write to the input registers. Address refers to the address of the input registers.

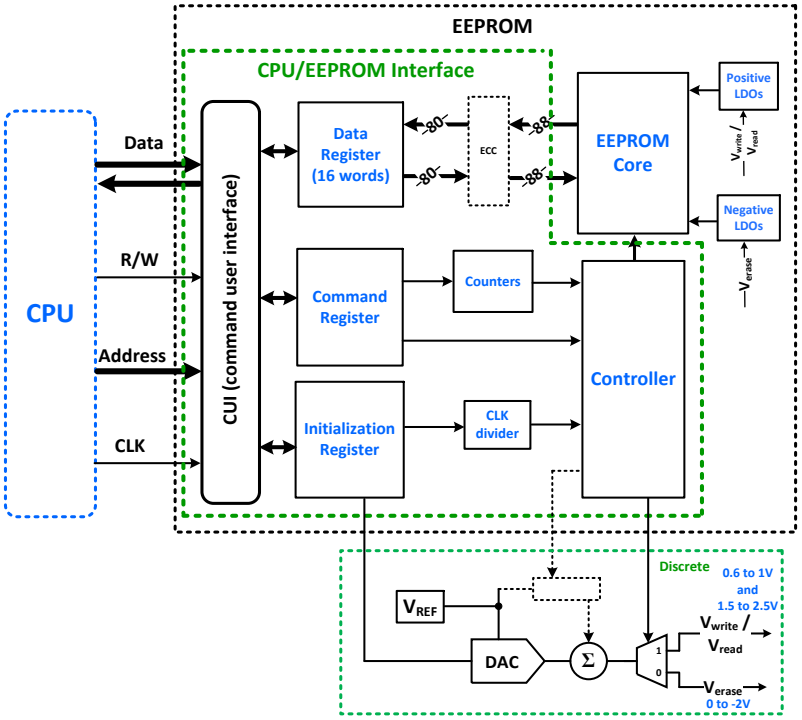


Figure 4.1 Top level block diagram of the EEPROM architecture with discrete programming control block.

4.2 CPU/EEPROM Interface

4.2.1 Input Registers

There are three input registers: initialization register, command register, and data register. The initialization register is 80 bits wide containing clock scaling value and DAC controlled programming voltage values for write, erase, and read operations respectively. Bit allocation of the initialization register is shown in figure 4.2. Each voltage value is 16 bits for supporting 16-bit DACs.

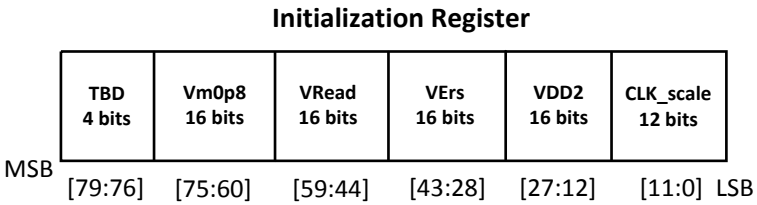


Figure 4.2 Initialization register bit allocation.

The data register is 16 words deep and 80 bits wide. It serves as a shared location for EEPROM and CPU to have mutually exclusive data communication. This means both EEPROM and CPU can read and write the data register but in a mutually exclusive manner. Data register outputs are expanded to 88 bits for supporting ECC function in the future version. Since CPU operates on a clock frequency much faster than the EEPROM, the data register is a data buffer for CPU to load and retrieve data after EEPROM processes it.

Command register is 80 bits wide and specifies EEPROM address, programming voltage stabilization time and application time for write, erase, and read operations, programming voltage quenching time, a BUSY signal, and memory operating commands. Figure 4.3 shows the bit allocation of the command register. The EEPROM address is specified by bits CMD [0:9] including word and bank address. The voltage stabilization time ensures the programming voltage stability at the output of the LDO before applying to the memory cells. Bits CMD [12:19], [28:35], and [44:51] specify the time duration for write, erase, and read voltage to become stable respectively. The voltage application time is the actual programming pulse time. Bits CMD [20:27], [36:43], and [52:59] specify the values for write, erase, and read respectively. The programming voltage quenching time, CMD [60:67], are provided as a waiting time for high programming voltage to switch off before engaging any state transition. The BUSY bit is used by the CUI to schedule data register access between CPU and EEPROM. When the BUSY bit is set, EEPROM can read from or write to the data register. When the BUSY bit is cleared, CPU regains access to the data register. This manner reduces CPU wait time and prevents data conflict. CMD

[72:79] specify memory operation, i.e. write, erase, and read. CMD [76:79] specify a block (16 words) operation when loaded with xF and a word operation when loaded with x0. Memory operation is specified by CMD [72:75]. Write (WR) operation corresponds to x1. Erase (ER) operation corresponds to x2. Read (RD) operation corresponds to x3. All other combinations of CMD [72:79] are considered as invalid for this design and reserved for future use. Bits [10:11] and [68:70] are reserved for future use and no hardware connections to them.

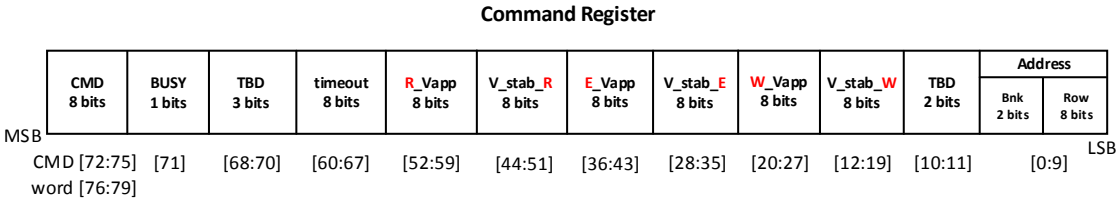


Figure 4.3 Command register bit allocation.

4.2.2 CUI

CUI, command-user interface, is a logic block that schedules data traffic among CPU, input registers, and EEPROM core. The CUI circuit diagram is shown in figure 4.4. This unit consists of multiplexers, a word counter and logic that steers data to the correct destination. The control signal is the BUSY bit. Three input registers mentioned in 4.2.1 have designated address specified by the two least significant bits of the CPU address, CPU_addr [0:1]. The CPURW bit indicates a CPU write to the input registers when it is set and read from them when cleared. The 16-word data register can be accessed by both CPU and EEPROM in a mutually exclusive fashion depending on the state of BUSY bit. When BUSY bit is reset, CPU has access to the data register. Whether write or read depends on CPURW bit status. When BUSY bit is set, EEPROM can either read from or write to the data register depending in the memory operation command. In this manner CPU and EEPROM can never access the data register simultaneously avoiding data conflict. The initialization register and command register may be read by the CPU at any time regardless of the status of the BUSY bit.

In addition to the control signals, a 4-bit word counter is used to track the operation progress through the data register. For a block of data, the word counter is loaded with value xF from the command register bit CMD [76:79]. For a signal word access, the word counter is loaded with value x0. During CPU access to the data register, the word counter is enabled when data register access is granted and decremented according to the CPU clock as data register access advances. During EEPROM access, the data register is read by the EEPROM under write or erase operation and written under EEPROM read operation. The word counter is decremented after each word access is complete indicated by either the signal tWR_done or tRD_done being true. During both CPU and EEPROM access, when the word counter finishes counting, the overflow bit, Cout, is set high indicating a completion of operation with data register. Cout signal is also used in the controller to determine the relevant states.

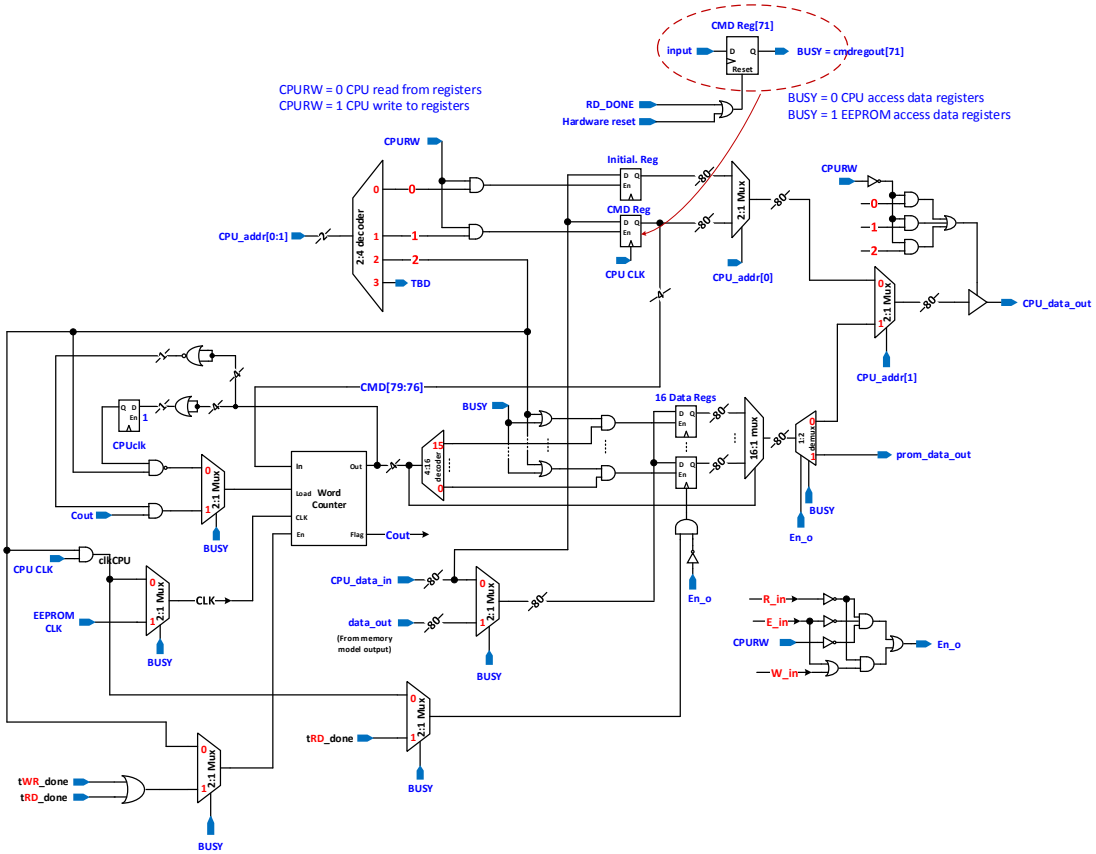


Figure 4.4 CUI (command-user interface) schematic.

4.2.3 Controller

The controller is a finite state machine controlling state advance throughout memory operation. Figure 4.5 shows the state diagram of the controller. Each state of operation has a designated timer, i.e. a counter. The quenching timer is shared between write and erase operation. The write voltage stabilization timer is loaded in the "Write Starts" state. This timer starts in the "VDD2 V_Stab" state. The write voltage application timer is loaded in "Write Starts" state or reloaded in the "W_done" state for next word write. The voltage application timer starts in "W_Vapp time" state. In "W_done" state, signals Cout and t_out are checked to determine whether control should move on to read operation or continue to write next word. Cout is the overflow bit of the word counter in the CUI. Logic high Cout bit means there is no more word to write. Otherwise there are words remaining to be written into the memory. t_out signal is the quenching timer indicator. It is logic high when quenching time is over. Therefore, combination of Cout = '1' and t_out = '1' shift control to next memory operation. And Cout = '0' and t_out = '1' continues the write operation. In our design, after write operation is complete, control enters memory read operation automatically to read out the words just written and store them back to the data register for CPU to retrieve and confirm.

The erase voltage stabilization timer is loaded in "Erase Starts" state and enabled in "VErs V_Stab" state. Erase voltage application timer is loaded in "Erase Starts" state and enabled in "E_Vapp time" state. This design erases a block of words at a time. Thus, the quenching timer starts at "E_done" state. After the quenching time, read operation starts automatically to confirm the words just erased. For next block of erase, CPU needs to wait CUI passes control from EEPROM to CPU to reload new erase command. The block access is consecutive from the first memory word location on a 16-word boundary. The starting block address must be integer multiples of 16. This design does not support block operations that overlaps between two memory banks.

Read voltage stabilization timer is loaded in “Read Starts” state and starts in “Vread V_Stab” state. Read voltage application timer is loaded in “Read Starts” state or reloaded in “R_done” state for a block word read. The controller also generates signals W_in, E_in, R_in, WR, ER, RD, and RD_DONE for control purpose. Signal W_in, E_in, and R_in maintain logic high from voltage stabilization state to x_done state of the respective operation. Signal WR, ER, and RD are true only at the voltage application state of corresponding operation. RD_DONE is generated when read operation is complete and used to reset the BUSY so that CUI returns data register access to the CPU.

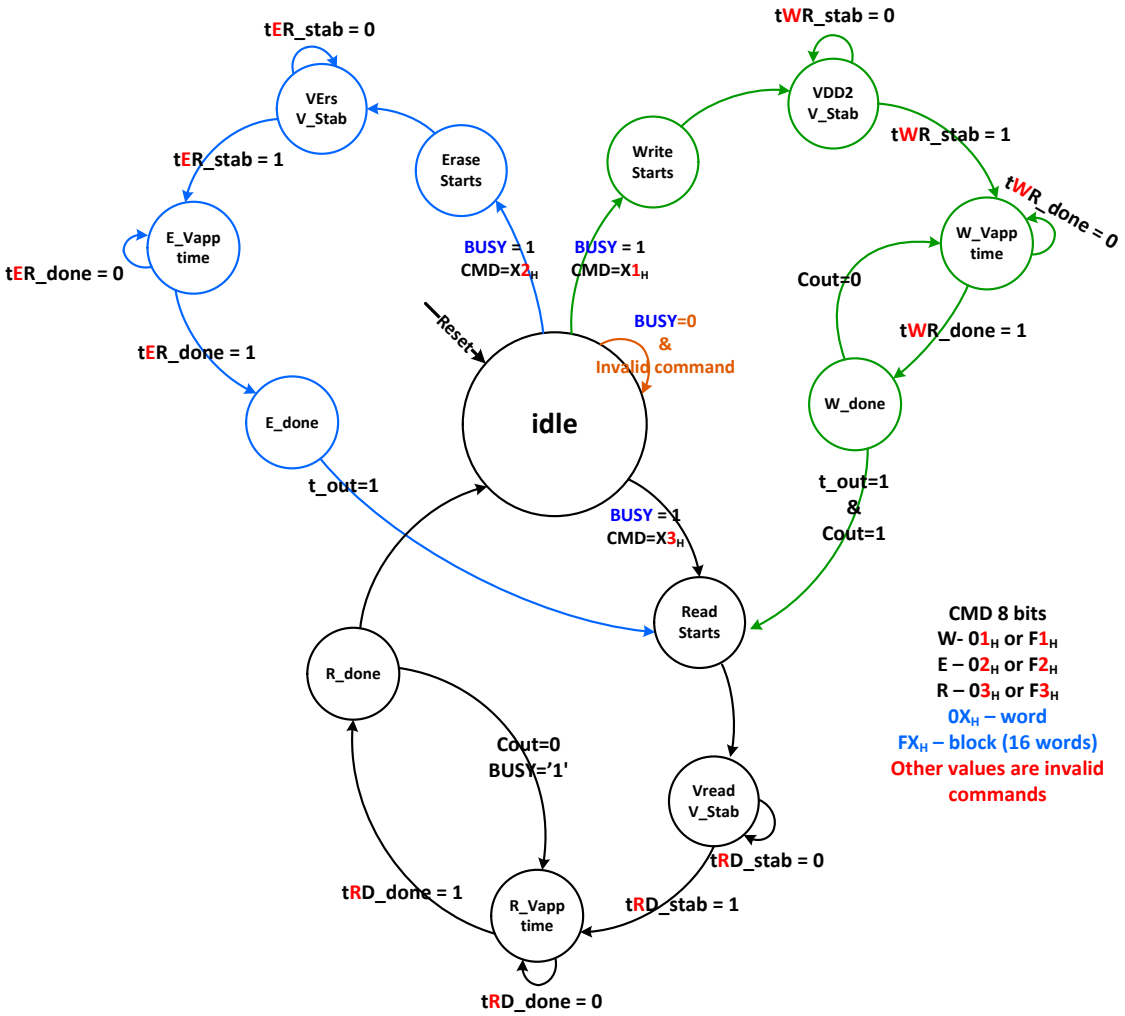


Figure 4.5 State diagram of the controller.

4.3 Interface Operation

4.3.1 Write

Upon starting a write operation the CPU loads the initialization, command, and data registers by inserting corresponding CPU address and true CPURW bit. CPU_addr [1:0] equals 00, 01, and 10 indicates initialization, command, and data registers respectively. CPU_addr = 11 is reserved for future use. When loading command register the first time, the BUSY bit (CMD[71]) should be loaded with 0 indicating CUI grands access to CPU. When CPU is ready to release access to EEPROM, command register is reloaded with true BUSY bit and other bits remains the same. This step makes the controller to start the EEPROM operation according to the command bits CMD [72:79]. The following shows example CPU command set for write operation.

CPURW <= '1';
CPU_addr <= "00"; -- initialization register
CPU_data_in <= x"123456789abcdef12340";
wait for 10 ns;
CPU_addr <= "01"; -- command register
CPU_data_in <= x"0100205020202020200 0"; -- BUSY bit is 0, CPU access data register.
wait for 10 ns;
CPU_addr <= "10"; -- data register
CPU_data_in <= x"a000000000000000000";
wait for 10 ns;
CPU_addr <= "01"; --reload command register
CPU_data_in <= x"0180205020202020200 0"; --BUSY bit is set to 1, EEPROM operation starts

Command register bits CMD [72:79] can be x01 or xF1 indicating write one word or a block of words respectively. When an invalid command is inserted, EEPROM control returns to idle state. At the same time CUI returns access to CPU. Command register address bit CMD [0:9] is decoded for accessing the target memory address and bank. When writing a block of words, only the first word's address is loaded to the address counter and the word counter in CUI is loaded

with xF. After a word is written, the address counter is incremented and word counter is decremented pointing to the next word in the data register. This process repeats until all 16 words in the data register are written to the memory. Read operation automatically starts after write operation. The words written will be read and stored in the data register for CPU to retrieve. After EEPROM read operation is completed, CUI returns access to CPU by resetting the BUSY bit. EEPROM control enters idle state. CPU can either recover the data register content or discard it by over writing with new value.

4.3.2 Erase

Erase operation is performed on a block of words. Erase command is xF2 at CMD [72:79]. Even though a block of words is erased at the same time, the bits CMD [76:79] need to be loaded with xF for the word counter to enable 16 words read after erase operation. Erase operation starts when the BUSY bit is set. Two separate load operations are necessary to the command register for setting up the word counter and BUSY signal. The starting address of the block to be erased is determined by decoding the command register bits CMD [4:7]. Read operation automatically starts after erase operation. The mechanism of CUI passing control from EEPROM to CPU is the same as write operation in 4.3.1. In order to erase another block, CPU needs to reload the command register with new address and erase command. The following is an erase command set example.

```
CPURW <= '1';
CPU_addr <= "00";    --No need to reload initialization register if its value is unchanged.
CPU_data_in <= x"123456789abcdef12340";
wait for 10 ns;
CPU_addr <= "01"; -- command register
CPU_data_in <= x"f200205020202020200"; -- BUSY bit is 0, CMD[76:79] loaded with f.
wait for 10 ns;
CPU_addr <= "01"; --reload command register
```

CPU_data_in <= x"f280205020202020200"; --BUSY bit is set to 1, EEPROM operation starts.

4.3.3 Read

Read operation command is CMD [72:79] equals xF3 or x03 indicating a block of words read or a single word read respectively. The command bits CMD [0:9] is decoded for target address and bank. Memory content at each address is stored in data register after read for CPU to retrieve. When reading a block of words, the starting memory address is decoded. Address counter increments after each word is read. The word counter decrements pointing to the next word to be stored into data register. After read operation is complete, BUSY signal is cleared returning data register access to CPU. EEPROM now enters idle state.

The following is a read command set example.

CPURW <= '1';
CPU_addr <= "00"; --No need to reload initialization register if its value is unchanged.
CPU_data_in <= x"123456789abcdef12340";
wait for 10 ns;
CPU_addr <= "01"; -- command register
CPU_data_in <= x"f300205020202020200"; -- read one block, BUSY bit is 0, CPU access data register.
wait for 10 ns;
CPU_addr <= "01"; --reload command register
CPU_data_in <= x"f380205020202020200"; --BUSY bit is set to 1, EEPROM operation starts.

4.4 EEPROM Core

4.4.1 EEPROM Cell

A differential cell structure is the most commonly used structure for differential current sensing. Research groups [23] and [56] have demonstrated their cell structure using CTT. Our work uses a similar but modified differential structure to improve programming control. Figure 4.6 shows the

cell structure along will controlling devices and control signals. One storage cell (or a bit) consists of logic device MC_1 and MC_2 forming the differential structure. The controlling devices include thick oxide high voltage devices M_1, M_2, MT_p, MT_n, and logic device Mn_1 and Mn_2. M_1 and M_2 control source voltage of the cell for different memory operations. MT_p and MT_n forms an inverter to control cell drain voltage. Mn_1 and Mn_2 resemble a switch for connecting/disconnecting cell to sense amplifier. Cell control signals include CS_Bk0 [0:87], CSbar_Bk0[0:87], N_Bk0 [0:87], Nbar_Bk0 [0:87], TL_Bk0 [0:87], and RS[0:255] for a bank. These signals are repeated for four banks according to the decoded bank address bits CMD [8:9]. This design has preserved the eight most significant bits of a word for ECC. The above cell control signals are generated according to the signals Bnk, WR, RD, ER, Data_ECC, which are generated by the controller. Storage cells have four operational states: low or no voltage idle state, low voltage read state, high voltage write state, and high voltage erase state. These four states are summarized in figure 4.7. Figure 4.7(a) depicts the idle state programming node voltage conditions. When the cell is in idle state, all word lines (RS[n]) are at zero volts. The drain of all the storage pair transistors is pulled to V_{SS} by the tail inverter composed by MT_p and MT_n. M_1 and M_2 are turned off. Write operation voltages are shown in figure 4.7(b). During write operation the drain of the storage pair is pulled to 1.5V by the tail inverter providing a source of current to generate hot carriers for trapping. The word line of the cell being written is supplied by write voltage, V_{write}. Either M_1 or M_2 will be turned on during write operation depending on the data applied to trap charges on one side of the differential cell. The side written with trapped charge will have positive threshold shift relative to the unwritten side. Figure 4.7(c) summarizes the erase voltage of the cell. The gate of the cell being erased can be supplied by a voltage in the range of 0V to -1.5V. The drain of the storage pair is 1.5V. Both M_1 and M_2 are turned on. This results in a reverse bias in the range of -1.5V to -3V begin applied to the storage transistors detrapping the carriers from the HK gate dielectric. Figure 4.7(d) shows the cell voltage condition for read operation. The gate voltage of the word being read is at V_{DD} and at V_{SS} for non-reading

words. The drain of the storage cell is pulled to V_{SS}. Both M_1 and M_2 are off. When the sense amplifier is enabled, i.e. M_{n1} and M_{n2} are turned on, current difference in the bit lines due to programmed threshold difference will be sensed to determine stored logic value.

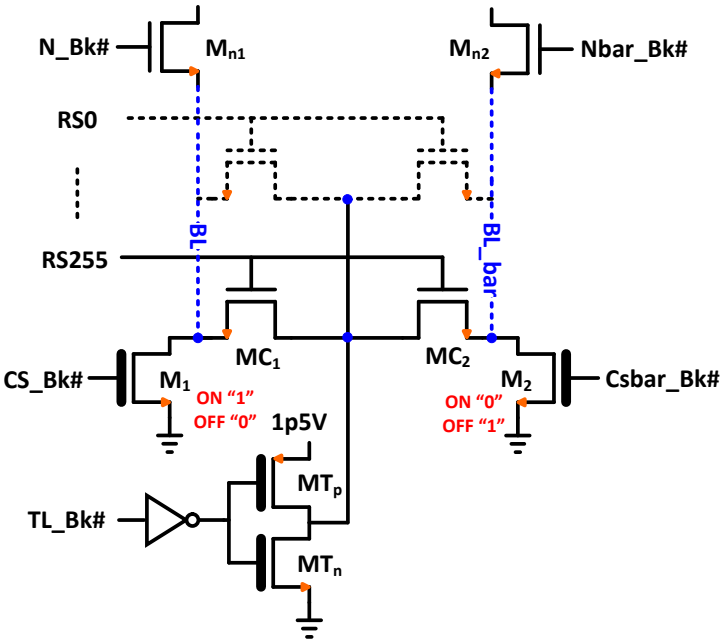


Figure 4.6 EEPROM cell with controlling device and control signals.

In order to ease the discussion of cell control signals generation, these signals are separated into two groups. One group of signals include CS_Bk0 [0:87], CSbar_Bk0[0:87], N_Bk0 [0:87], Nbar_Bk0 [0:87], and TL_Bk0 [0:87]. The Bk# in the signal name refers to a bank. For three other banks, the names are Bk1, Bk2, and Bk3. Generation of these signals depends on decoded bank select signal Bnk, WR, RD, ER, and Data_ECC. Table 4.1 summarizes signal status for each memory operation. One important point is that the side of the differential storage cell trapping carriers has higher threshold voltage relative to the adjacent side. Higher threshold means less current is sensed. Thus, it results logic zero when read by the sense amplifier.

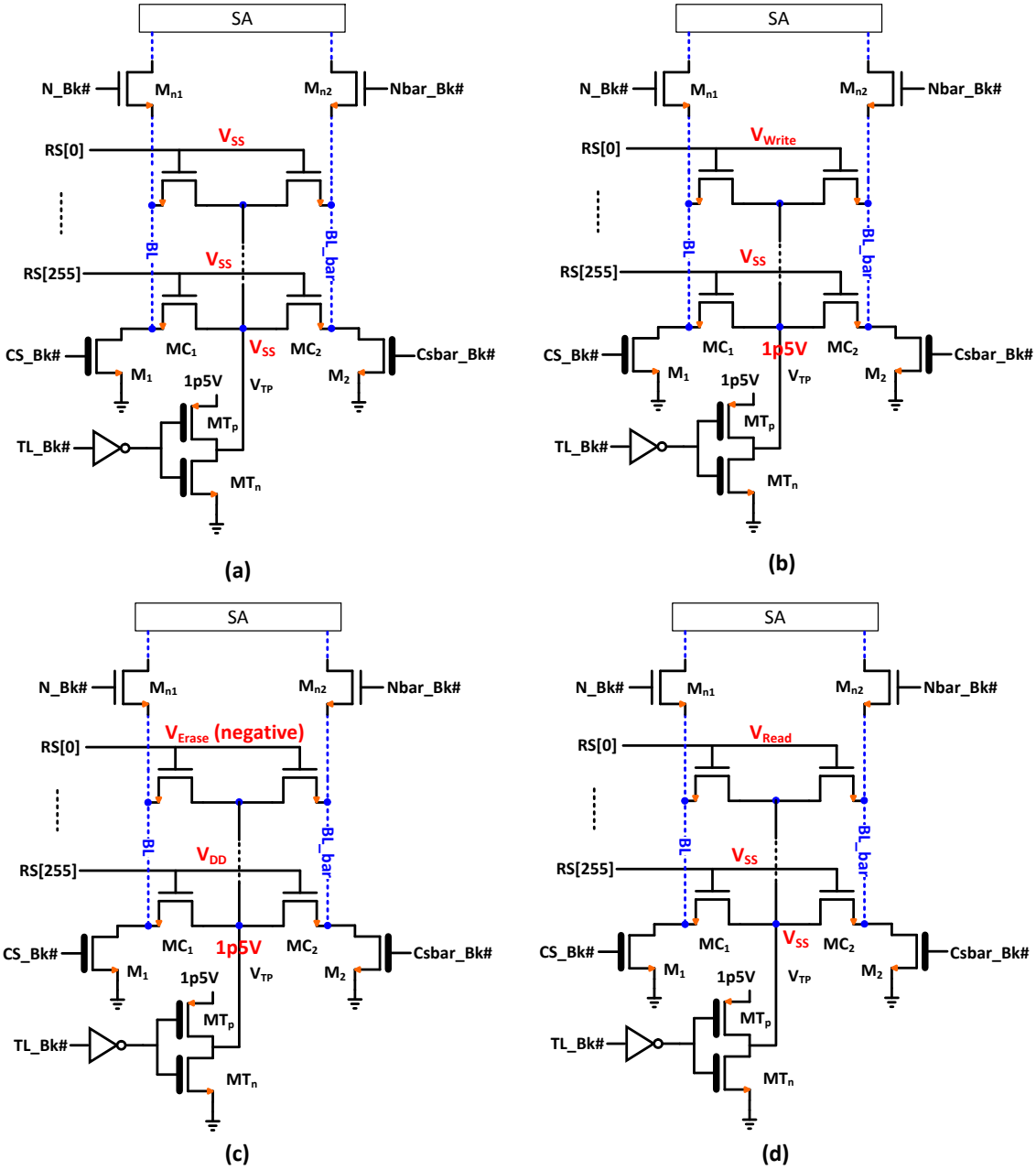


Figure 4.7 Circuit of a column of storage cell with sense amplifier block. (a) Idle state control voltage conditions. (b) Write operation voltage conditions. (c) Erase operation voltage conditions. (d) Read operation voltage conditions.

Table 4.1 Signal status summary of each memory operation.

	Signals	Write **WR**=1, **ER**=0, **RD**=0	Erase **WR**=0, **ER**=1, **RD**=0	Read **WR**=0, **ER**=0, **RD**=1
Selected bank **Bnk(#)** = 1	**CS_Bk#**	**Data_ECC**	0	0
	CSbar_Bk#	**Data_ECC_bar**	0	0
	N_Bk#	**Data_ECC_bar**	1	1
	Nbar_Bk#	**Data_ECC**	1	1
	TL_Bk#	1	1	0
Non-selected bank **Bnk(#)** = 0	**CS_Bk#**	0	1	0
	CSbar_Bk#	0	1	0
	N_Bk#	1	0	0
	Nbar_Bk#	1	0	0
	TL_Bk#	1	0	0

The other group of signals are the 256 row select (RS[#]) signals which are generated by logic signal pside[#] and nside[#] via level shifter shown in figure 4.8. RS[#] signals supply programming voltages to the gates of all storage transistors. Write and read voltages are supplied by PMOS controlled by pside[#] through p-side level shifter. Erase voltage is supplied by NMOS controlled by nside[#] through n-side level shifter. Signal status of pside[#] and nside[#] for memory operations are derived from level shifter logic. Table 4.2 summarizes pside[#], nside[#], and RS[#] signal status for each memory operation. Signal pside[#] and nside[#] are generated based on signals WR, ER, RD, decoded address signal addr_decode, and block select signal Blk, which is decoded address bits CMD [4:7]. Figure 4.9 shows the interface logic circuits. CUI circuit is given by figure 4.4.

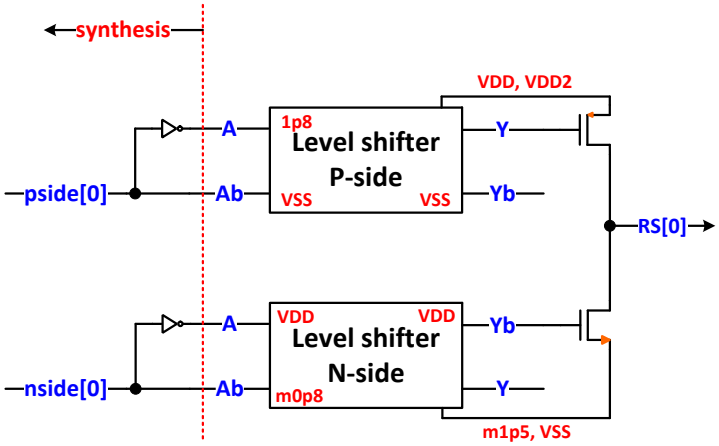


Figure 4.8 RS[#] signals generation by pside[#] and nside[#] via level shifters.

Table 4.2 RS[#] signals summary for each memory operation.

	pside	**nside**	**RS**
Write (**WR**=1, **ER**=0, **RD**=0, **addr_decode**=1, **Blk**=0)	1	0	VDD2
Non write (**WR**=1, **ER**=0, **RD**=0, **addr_decode**=0, **Blk**=0)	0	1	VSS
Erase (**WR**=0, **ER**=1, **RD**=0, **addr_decode**=x, **Blk**=1)	0	1	m1p5
Non erase (**WR**=0, **ER**=1, **RD**=0, **addr_decode**=x, **Blk**=0)	1	0	VDD
Read (**WR**=0, **ER**=0, **RD**=1, **addr_decode**=1, **Blk**=0)	1	0	VDD
Non read (**WR**=0, **ER**=0, **RD**=1, **addr_decode**=0, **Blk**=0)	0	1	VSS
Idle (**WR**=0, **ER**=0, **RD**=0, **addr_decode**=0, **Blk**=0)	0	1	VSS

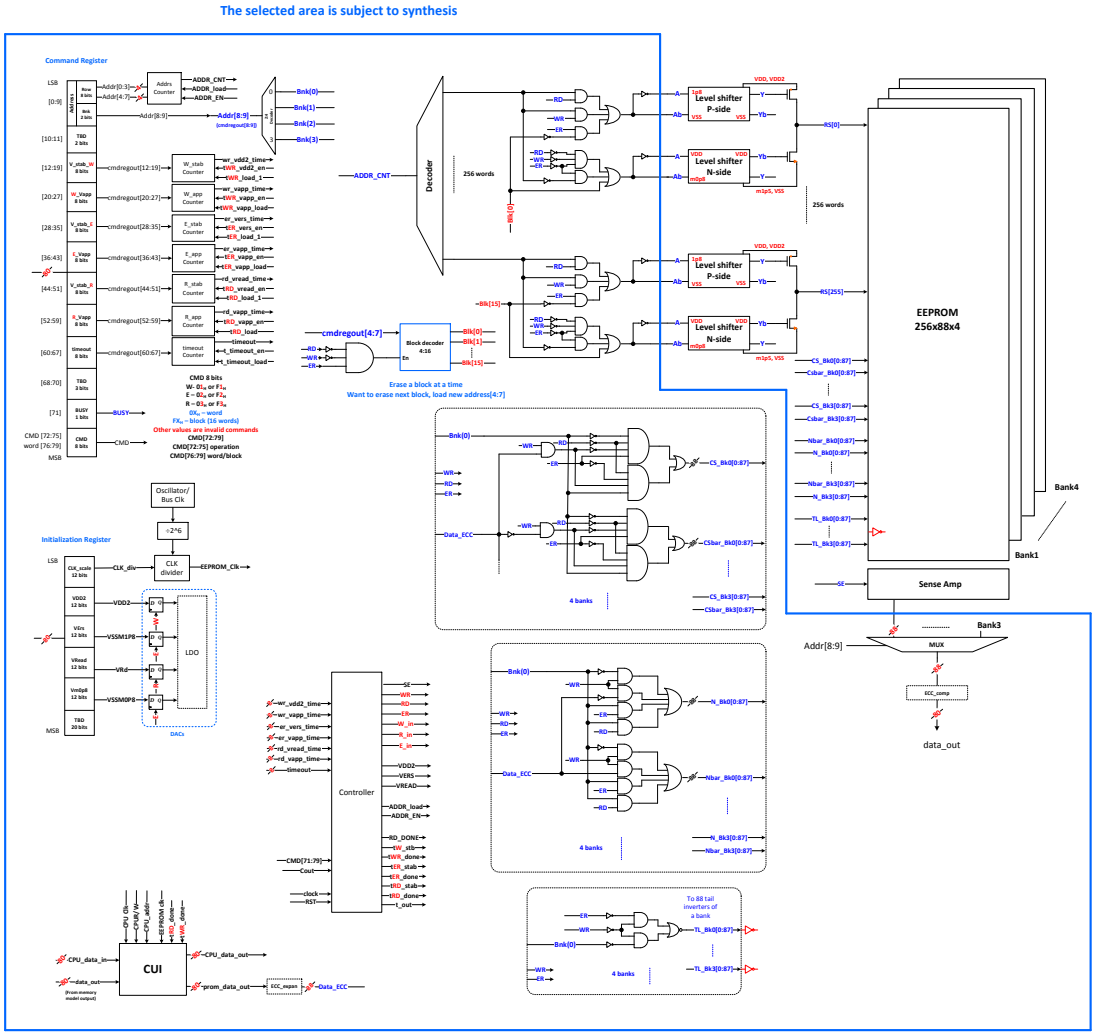


Figure 4.9 Full CPU/EEPROM interface schematic.

4.4.2 EEPROM Macro Model

In order to verify the logic correctness of design the interface, the EEPROM core is modeled by a 256x88 bits array. This array is repeated four times mimicking four banks. Each array is controlled by signals CS_Bk#[0:87], CSbar_Bk#[0:87], N_Bk#[0:87], Nbar_Bk#[0:87], and TL_Bk#[0:87]. The following code describes the control signal status for write, erase, and read operations.

state_bk0 <=

w_bk0 when

```
(
        (CS_bk0 /= CSbar_bk0) and (N_bk0 /= Nbar_bk0) and
        (TL_bk0 = x"00000000000000000000")
)else
e_bk0 when
(
        (CS_bk0 = x"00000000000000000000") and
        (CSbar_bk0 = x"00000000000000000000") and
        (N_bk0 = x"ffffffffffffffffffff") and
        (Nbar_bk0 = x"ffffffffffffffffffff") and
        (TL_bk0 = x"00000000000000000000")
)else
r_bk0 when
(
        (CS_bk0 = x"00000000000000000000") and
        (CSbar_bk0 = x"00000000000000000000") and
        (N_bk0 = x"ffffffffffffffffffff") and
        (Nbar_bk0 = x"ffffffffffffffffffff") and
        (TL_bk0 = x"ffffffffffffffffffff")
);
```

The above code is repeated for four banks. When write operation signal conditions are met, w_bk# is high. When erase operation signal conditions are met, e_bk# is high. When read operation signal conditions are met, r_bk# is high. The modeled array is written, read, and erased based on w_bk#, r_bk#, and e_bk# status respectively. The following code shows memory operation based on the control signals. Also, this part of the code is repeated four times for four banks.

```
prom_bk_one: process(data_ecc, ADDR_CNT, n_encode, clock)
begin
        if(clock'event and clock='1') then
                if(state_bk0 = w_bk0) then
```

```
                    prom_bk0(to_integer(unsigned(ADDR_CNT))) <= data_ecc;
            elsif(state_bk0 = r_bk0) then
                    data_out_ecc_bk0 <= prom_bk0(to_integer(unsigned(ADDR_CNT)));
                    databar_out_ecc_bk0 <= not
prom_bk0(to_integer(unsigned(ADDR_CNT)));
            elsif(state_bk0 = e_bk0) then
                    for i in 0 to 15 loop
                            prom_bk0((to_integer(unsigned(n_encode)))+i) <=
(others=>'0');
                    end loop;
            end if;
        end if;
end process;
```

CHAPTER V

PROGRAMMING VOLTAGE GENERATION

This chapter discusses low dropout voltage regulator (LDO) implementation for generating the programming voltages. In order to have highly accurate control over the programming voltages, on-chip LDO and off-chip DACs are employed. The DAC controlled LDO provide fine tuning to the programming voltages avoiding catastrophic damage to the device due to sudden high voltages. The following sections discuss LDO specifications, design flow, and implementation.

5.1 LDO Specifications

Parameters that matter to this application are minimum programming voltage step, LDO output current and voltage control, LDO and DAC accuracy, and LDO stability. The LDO settling time is not a burden in this application since we have allocated voltage stabilization time in the interface that provides enough time for the programming voltage to stabilize. With the given process, the 10μs designed settling time should easily be met. Khan et al. [22] [55] have suggested 10ms programming pulse with 10mV and 50mV increments. We decided to use 5mV programming step to achieve fine tuning safety margin on the programming voltages. Two 16 bits discrete DACs are selected (DAC7631 and LTC1650). The LDO output swing is from 0.6V to 2.5V which covers read and write operations. The erase voltage is generated by a similar LDO which has negative output. The erase LDO output is negative 1.5V that results a negative 3V across the storage cell.

The main current load to the LDO is storage cell gate leakage of one word. The programming current of a word is much less compared to the leakage. Fowler-Nordheim tunneling is used to estimate the programming current. Restate FN tunneling equation (3.4) here,

$$J_{nFN} = A_{nFN} E_{ox}^2 e^{-\frac{B_{nFN}}{E_{ox}}} \tag{5.1}$$

where A_{nFN} = 6.55x10^{-6} A/V^2, B_{nFN} = 2.85x10^8 V/cm [73], and $E_{ox} = \frac{V_g - V_{FB}}{t_{ox}}$.

Threshold voltage is [74]

$$V_{th} = V_{FB} + 2\phi_f + \frac{\sqrt{4qN_a\varepsilon_{Si}\phi_f}}{C_{ox}} \tag{5.2}$$

Substituting (5.2) into (5.1), FN current density can be approximated as

$$J_{nFN} \approx A_{nFN}\left(\frac{V_g - V_{th}}{t_{ox}}\right)^2 e^{-\frac{B_{nFN}}{\frac{V_g - V_{th}}{t_{ox}}}} \tag{5.3}$$

With 2V write voltage, ~1nm thick IL, and ~0.4V threshold voltage, FN tunneling current density of one transistor is about

$$J_{nFN} = (6.55 \times 10^{-6} A/V^2)\left(\frac{2V - 0.4V}{1nm}\right)^2 e^{-\frac{\left(2.85\times\frac{10^8 V}{cm}\right)(1nm)}{2V - 0.4V}} = 308043\ A/m^2 \tag{5.4}$$

One cell transistor is 650nm x 40nm. FN tunneling current of one word is

$$I_{nFN} = J_{nFN}\ x\ Area = 308043A/m^2\ x\ 650nm\ x\ 40nm\ x\ 88 = 704nA \tag{5.5}$$

Using the typical HVTNFET (high voltage thick NMOS) off current, I_{OFF}, form PDK which is 94nA/μm, leakage current of one word is calculated to be about 10μA. The LDO is designed for 2mA load current for margins.

Accuracy of the LDO is given by

$$Accuracy\% \approx \frac{\Delta V_{IR}+\Delta V_{LR}+\sqrt{\Delta V_{o,REF}^2+\Delta V_{O,amp}^2+\Delta V_{O,R}^2+\Delta V_{TC}^2}}{V_o} \times 100\% \quad (5.6)$$

where ΔV_{IR} is line regulation, ΔV_{LR} is load regulation, $\Delta V_{o,REF}$ is voltage reference drift, $\Delta V_{o,amp}$ is error amplifier drift, $\Delta V_{o,R}$ is feedback resistor network tolerance and ΔV_{TC} is temperature coefficient $\Delta V_{TC} = TC \cdot (T_{max} - T_{min})V_o$. Expanding quantities ΔV_{IR} and ΔV_{LR}, equation (5.7) is obtained.

$$Accuracy\% \approx \frac{\frac{1}{A\cdot\beta}+\frac{R_1+R_2}{g_{m,pass}\cdot A\cdot R_1}+\sqrt{\Delta V_{o,REF}^2+\Delta V_{O,amp}^2+\Delta V_{O,R}^2+\Delta V_{TC}^2}}{V_o} \times 100\% \quad (5.7)$$

where A is error amplifier open loop gain, feedback factor β equals $R_I/(R_I + R_F)$ shown in figure 5.1. To simplify the calculation, we assume the square root term in equation (5.7) is less than the sum of other terms. Since the DAC (DAC7631) has full scale error of ±1mV, we set 0.33% accuracy of the LDO to preserve the DAC accuracy when LDO output is 0.6V. The open loop gain of the error amplifier is obtained using equation (5.7) and is about 133dB.

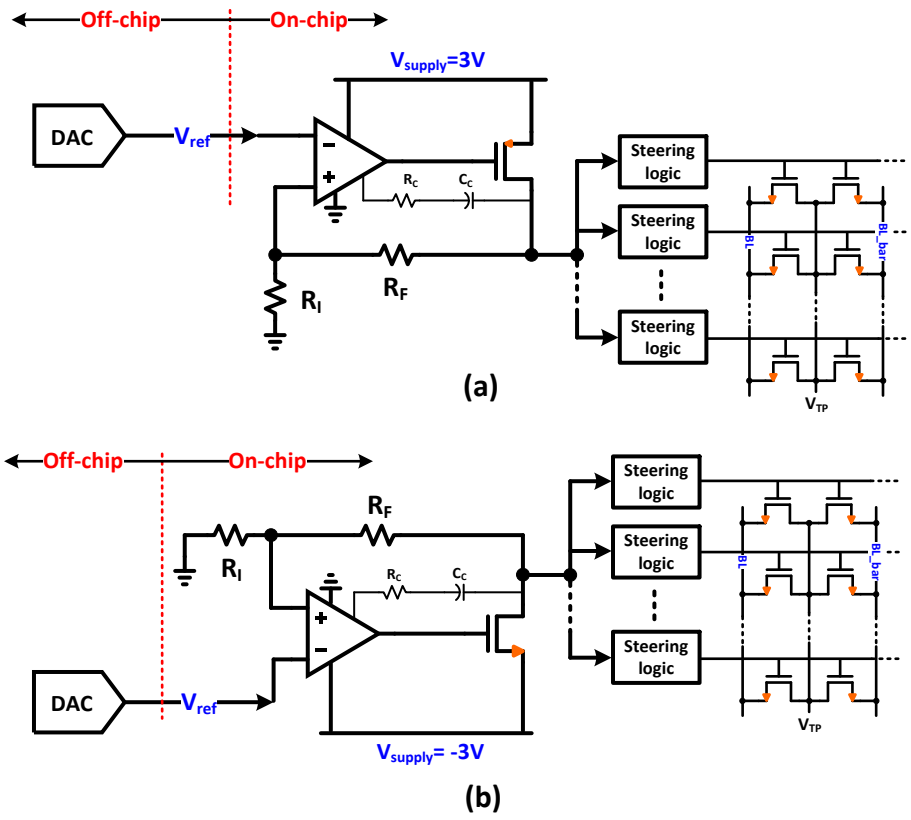
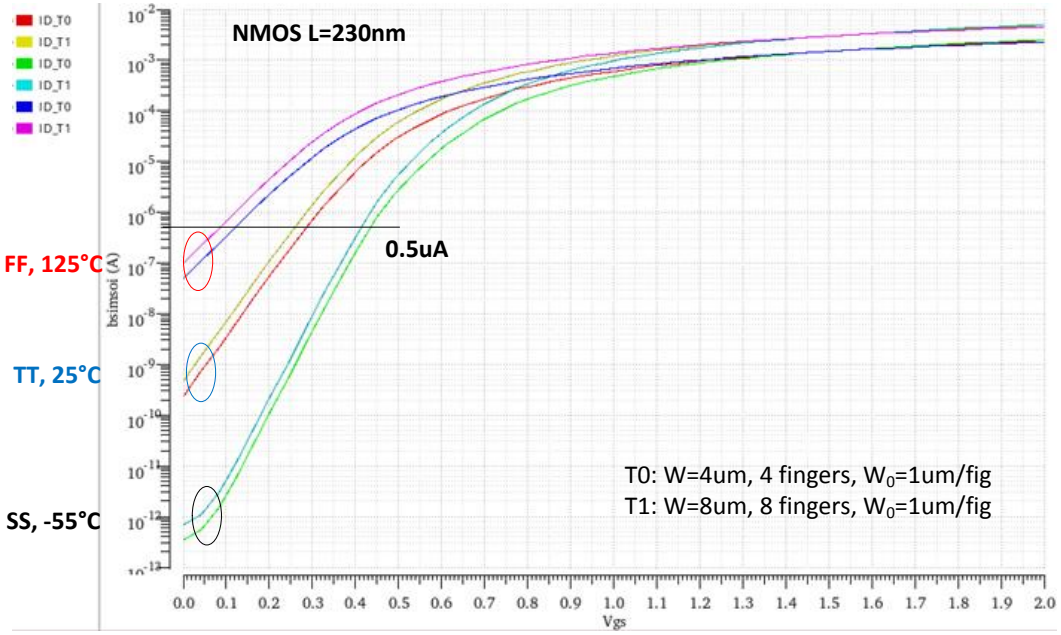


Figure 5.1 On-chip LDO and off-chip DAC for generating EEPROM programming voltages. (a) LDO for write and read voltage. (b) LDO for erase voltage.

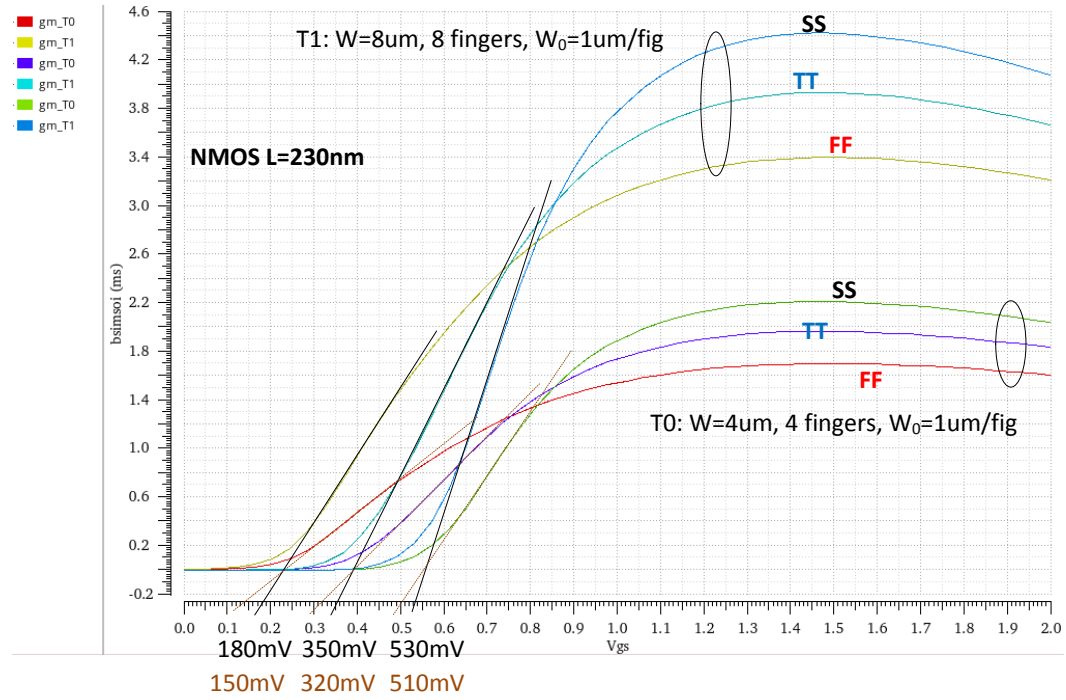
5.2 Design Flow

Design begins by knowing individual transistors performance capability of different fingers for a given process. The performance parameters include transconductance, g_m, output transconductance, g_{ds}, intrinsic gain, μ, and unity current gain bandwidth, f_{TA}. These parameters are evaluated with respect to different bias currents and finger count. Analog applications require that transistors operate in saturation region where they behave as current sources. Therefore, it is efficient to work with these parameters in terms of current density. Designing in terms of current density allows devices track each other across process and temperature variations because each device is formed as a composite identical unit, a finger. Additionally, most designs can be scaled

by changing finger ratios. The individual transistor performance is studied via parametric analysis.



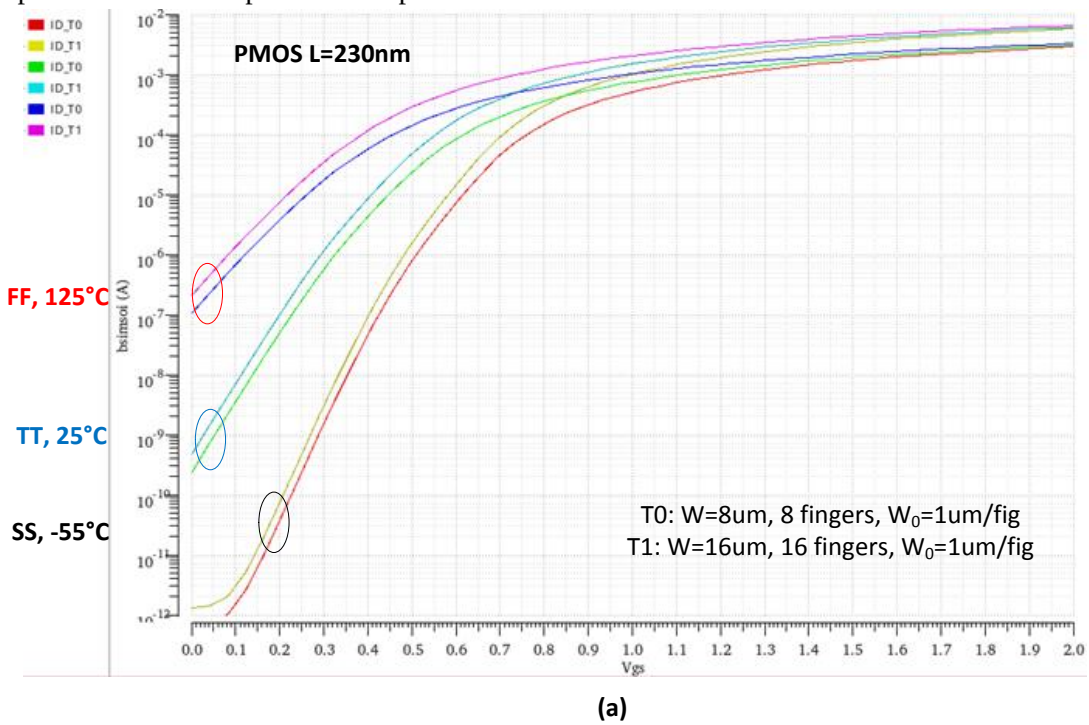
(a)



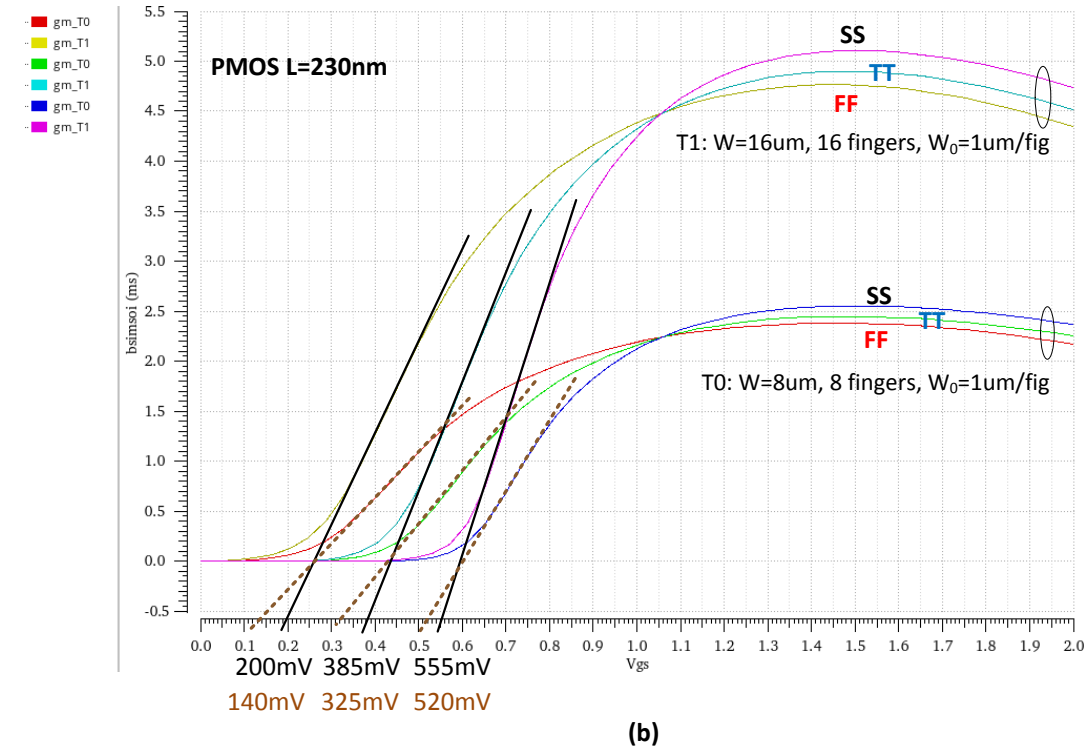
(b)

Figure 5.2 L=230nm thick oxide NMOS. (a) I_d-V_{gs} plot of 4 finger and 8 finger devices across temperature and process corners. (b) g_m-V_{gs} plot of 4 finger and 8 finger devices threshold

extrapolation across temperature and process corners.



(a)



(b)

Figure 5.3 L=230nm thick oxide PMOS. (a) I_d-V_{gs} plot of 8 finger and 16 finger devices across temperature and process corners. (b) g_m-V_{gs} plot of 8 finger and 16 finger devices threshold extrapolation across temperature and process corners.

Several NMOS and PMOS devices with different finger numbers can be investigated. A typical number of fingers such as 1, 4, 8, and 16 can be used as a starting point. It is important to notice that all transistors have the same length and width for a unit finger. The parameters mentioned above are plotted with respected to different bias currents. Bandwidth and extreme settling time are not required for this application. For this reason, the LDO is designed to operate in subthreshold region to minimize power consumption. Figure 5.2(a) shows the I_d-V_{gs} plot of thick oxide NMOS (L=230nm) across temperature and process corners for 4 finger and 8 finger respectively. Figure 5.2(b) shows the threshold extrapolation for the same device geometry across temperature and process corners. The purpose of these plots is to determine the overdrive voltage of the device. Similarly, figure 5.3 shows the same plots for the PMOS. Figure 5.4 shows the g_m, g_{ds}, self-gain, and f_{TA} values of typical process corner 8um wide NMOS and 16um wide PMOS biased at 0.5µA.

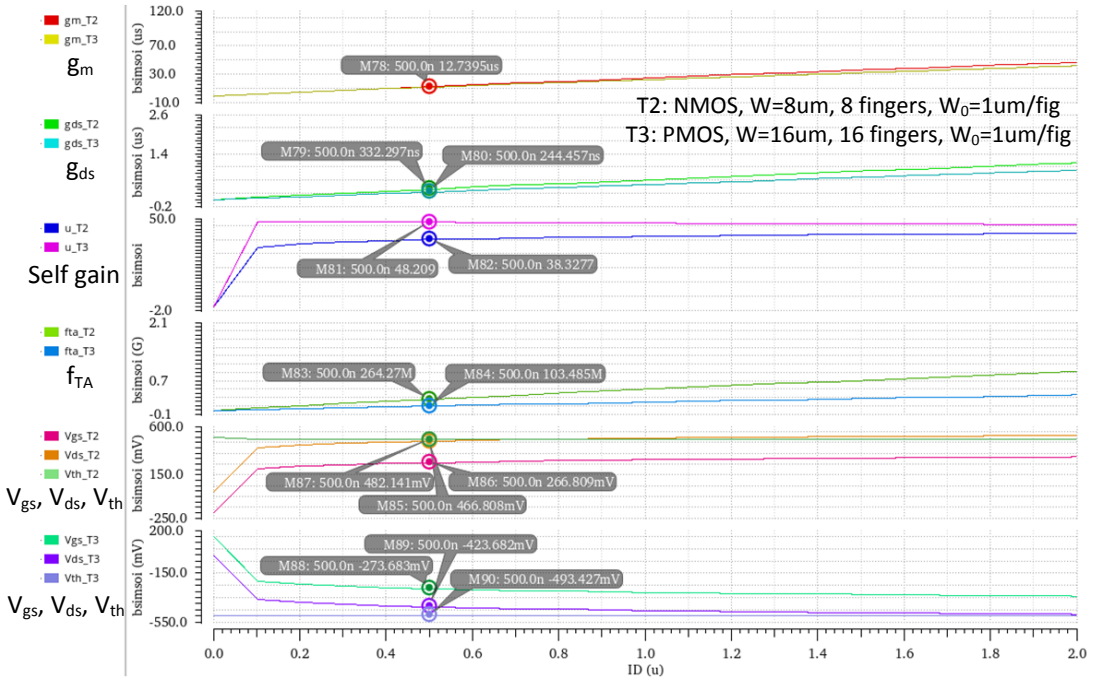


Figure 5.4 g_m, g_{ds}, self gain, f_{TA}, V_{gs}, V_{ds}, and V_{th} plots verse I_d of 8um wide NMOS and 16um wide PMOS at typical-typical process corner respectively.

After knowing the "unit" transistor performance capability, one can proceed with circuit design. An LDO design starts by knowing the maximum output load current requirement. As discussed in section 5.1, LDO in this work is to provide gate programming voltage and current about 2.5V and 704nA respectively. For erase operation, if one word erase current is assumed to be about the same as write, a block of word erase current is about 12μA. The LDO is designed to support 2mA current for enough margin. Per the accuracy requirement stated in section 5.1, the error amplifier ought to have high gain of 133dB for. This high gain leads to the telescopic topology with gain boost on the cascode p- and n-transistors for the error amplifier. Circuit stability is another critical design factor. The potential stability issue of LDO appears at minimum load or no load current conditions. This issue arises from the output pole shift due to load current varies. At minimum load or no load current situations, the output pole has the potential to fall below the GBP of the OTA. If this occurs, there would not be enough phase margins to maintain closed-loop stability. As the result, oscillation, ringing or poor settling take place in the transient response. This issue can be visualized by using a Bode magnitude plot shown in figure 5.5. In order to maintain

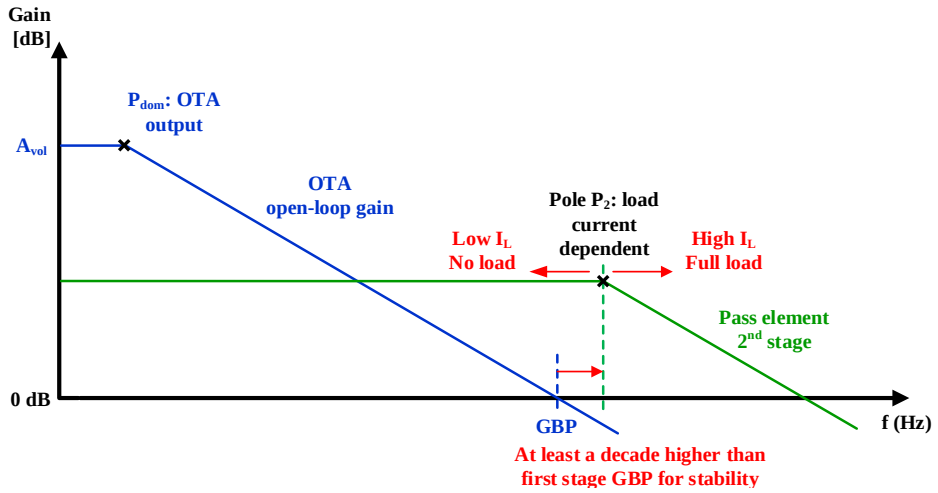


Figure 5.5 Output pole location as a function of load current relative to the GBP of the OTA.

stability, lead-lag compensation is used to further split the dominant pole and the first nondominant plot so that there is only the dominant pole present before unity gain frequency.

Figure 5.6 shows the closed-loop small signal LDO circuit for stability analysis. In the figure 5.6 (a), $g_{m,ota}$ is the transconductance of the error amplifier. $C_{para,ota}$ and $r_{o,ota}$ are error amplifier's output capacitance and resistance respectively. The pass PMOS transistor is shown in its small signal circuit model. R_F and R_I compose the feedback resistor network. The closed-loop transfer function is given as

$$\frac{v_o}{v_{ref}} = \frac{g_{m,ota}r_{o,ota}^2(R_1+R_2)(g_{m,pa}-C_{gd,pa}s)}{C_L r_{o,ota}s(R_1+R_2)+C_{gd,pa}r_{o,ota}s\left(R_2-R_1\left(g_{m,ota}r_{o,ota}-1\right)\right)+R_1\left(g_{m,ota}g_{m,pa}r_{o,ota}^2+1\right)+R_2+r_{o,ota}} \quad (5.8)$$

The output pole can be solved and simplified to

$$s_{output} = -\frac{1}{C_L(R_1+R_2)+C_{gd,pa}(R_2-R_1 g_{m,ota}r_{o,ota})} \quad (5.9)$$

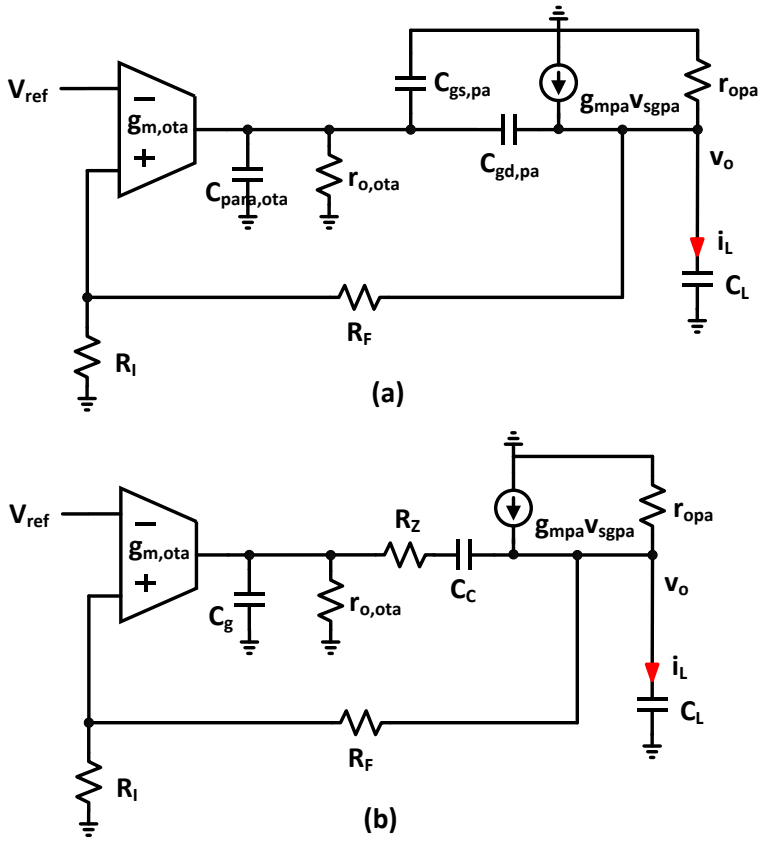


Figure 5.6 (a) LDO closed-loop small signal equivalent circuit with simplified error amplifier (OTA). (b) LDO closed-loop small signal equivalent circuit with lead-lag compensation.

The minimum output current is determined by minimum output voltage and summation resistance of feedback resistor network and a keeper current to maintain stability. For 500Ω R2, 1kΩ R1 and 0.6V output voltage, the minimum output current is 0.4mA. For 2.5V output voltage, the maximum output current is 1.67mA. From simulation, at minimum current condition, $g_{m,ota}$ = 10.7uS, $r_{o,ota}$ = 51.9GΩ, $C_{gd,pa}$ = 39.9fF. Using equation (5.9), the output pole is at about 7.16 kHz. This frequency is much lower than the 118 MHz GBP of the OTA which is calculated by

$$GBP_{ota} = \frac{g_{m,ota}}{2\pi C_{gs,pa}} \tag{5.10}$$

where $C_{gs,\,pa}$ is the gate capacitance of the pass PMOS and equals 89fF. Therefore, lead-lag compensation is used to maintain stability.

Figure 5.6 (b) shows the LDO closed-loop small signal circuit with lead-lag compensation. In figure 5.6(b), C_g is the sum of $C_{para,ota}$ and $C_{gs,\,pa}$. C_C is the compensation capacitor. R_Z is the nulling resistor to cancel the effect due to right-half plane zero raised from feedforward current in C_C. The right-half plane zero is at frequency

$$\omega_{zero,RHP} = \frac{g_{m,pa}}{C_C} \tag{5.11}$$

by adding the null resistor, the zero frequency changes to

$$\omega_{zero,null} \approx -\frac{1}{C_C\left(R_Z - \frac{1}{g_{m,pa}}\right)} \tag{5.12}$$

If R_Z equals $\frac{1}{g_{m,pa}}$, the RHP zero is pushed to infinity. If R_Z is greater than $\frac{1}{g_{m,pa}}$, the RHP zero is converted to lift-half plane. The effect of compensation can be seen from figure 5.7. The compensation capacitor reduces the dominant pole to lower frequency and pushes the first nondominant pole to higher frequency ensuring loop gain approaches unity gain frequency at the rate of -20dB/dec while keeping all RHP zeros at higher frequencies.

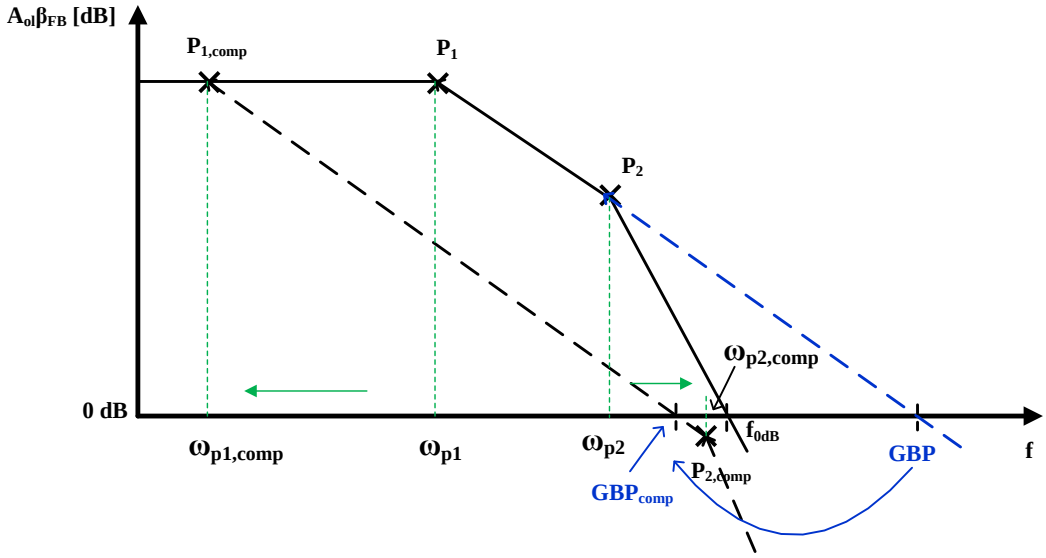


Figure 5.7 Lead-lag compensation achieves pole splitting for closed-loop stability.

5.3 Implementation

5.3.1 Bandgap Voltage Reference

The LDO uses bandgap voltage reference as bias generator to generate bias current. The basic idea of bandgap voltage reference circuit is the mutual compensation between PTAT term (proportional to absolute temperature) and CTAT (complementary to absolute temperature) to achieve a reference voltage or current (this work) with zero temperature coefficient [74]. Figure 5.8 shows the circuit diagram. As mentioned above, this design is for subthreshold operation. Drain current is given as

$$I_D = \mu C_{ox}(n-1)\frac{W}{L}U_T^2 e^{\frac{V_{GS}-V_{th}}{nU_T}} \tag{5.13}$$

where μ is mobility, C_{ox} is gate capacitance, n is subthreshold slope, W and L are transistor width and length, U_T is thermal voltage. One can write an equation for the loop that consists of transistor M_1, amplifier, R_2, and transistor M_2 as

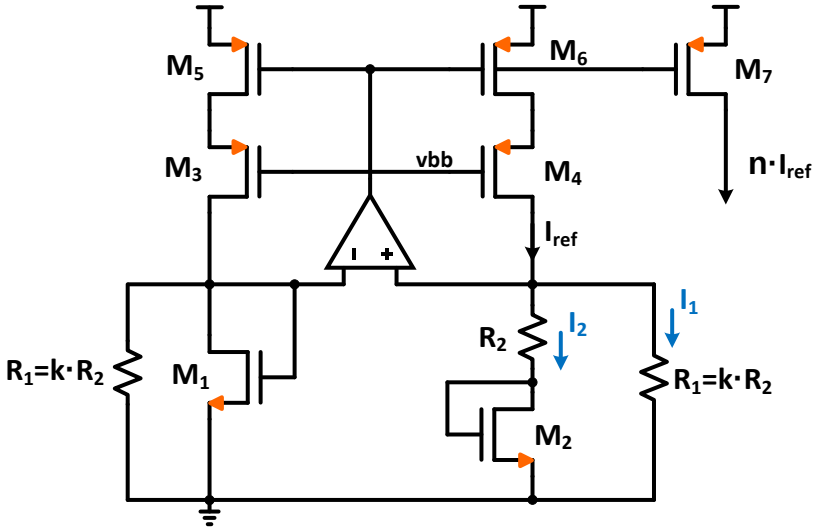


Figure 5.8 Bandgap voltage reference circuit diagram.

$$V_{GS1} = I_2 R_2 + V_{GS2} \tag{5.14}$$

Equation (5.13) and (5.14) can be used to solve current I_2,

$$I_2 = \frac{nU_T \ln(S)}{R_2} \tag{5.15}$$

where S is the transistor width ratio of M_2 and M_1. As shown by figure 5.8, reference current is the sum of I_2 and I_1 and can be written as

$$I_{ref} = \frac{nU_T \ln(S)}{R_2} + \frac{I_2 R_2 + V_{GS2}}{R_1} \tag{5.16}$$

By substituting $U_T = \frac{kT}{q}$ into equation (5.16),

$$I_{ref} = \frac{nkT \ln(S)}{qR_2} + \frac{nkT \ln(S)}{qR_1} + \frac{V_{GS2}}{R_1} \tag{5.17}$$

By taking derivative of equation (5.17) with respect to temperature, one can obtain

$$\frac{\partial I_{ref}}{\partial T} = \frac{nk \ln(S)}{qR_2} + \frac{1}{R_1}\left(\frac{\mathrm{nk}\ln(S)}{q} + \frac{\partial V_{GS2}}{\partial T}\right) \tag{5.18}$$

By setting equation (5.18) to zero and substituting n=1.4, $\frac{k}{q}$ = 0.085mV/°C, S=4, and $\frac{\partial V_{GS2}}{\partial T}$ = -1.165 mV/°C from simulation for the corresponding finger number, resistor ratio between R_2 and R_1 can be found as

$$\frac{\partial I_{ref}}{\partial T} = \frac{1}{R_2}(1.4)\left(\frac{0.085mV}{°C}\right)(1.39) + \frac{1}{R_1}\left(-\frac{1.165mV}{°C} + 1.4 \times \frac{0.085mV}{°C} \times 1.39\right) = 0 \qquad (5.19)$$

Therefore,

$$\frac{R_1}{R_2} = 6.05 \qquad (5.20)$$

R_2 is found using equation (5.16) and (5.20) where V_{GS2} is taken from I_D-V_{GS} characteristic curve. After reference current, I_{ref}, is determined, a scaled version can be distributed to the reset of the circuit by adjusting transistor width ratio between M_6 and M_7 in figure 5.8.

5.3.2 Gain Boosting

The gain boosting technique increases voltage gain by increasing output impedance without adding more cascode devices [114]. Figure 5.9 shows the circuit topology and its small signal circuit for output resistance analysis.

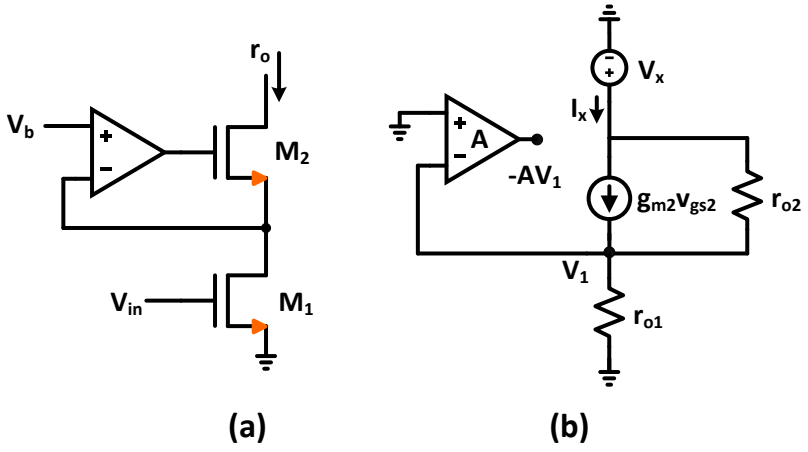


Figure 5.9 (a) Gain boosting circuit topology [114]. (b) Small signal circuit of (a) for output resistance analysis.

By writing output node current equation

$$-I_x + g_{m2}v_{gs2} + \frac{V_x - V_1}{r_{o2}} = 0 \tag{5.21}$$

and knowing the relation

$$V_1 = I_x r_{o1} \tag{5.22}$$

output resistance can be solved as

$$r_o = r_{o1} + r_{o2} + g_{m2}r_{o1}r_{o2}A \approx g_{m2}r_{o1}r_{o2}A \tag{5.23}$$

The output resistance is boosted by the gain of the amplifier.

5.3.3 Boosted Telescopic OTA

To achieve high open-loop gain, a boosted single ended telescopic OTA is used as the error amplifier. Figure 5.10 shows the circuit topology. The boosting amplifier has gain of

$$G_{n\ side} = g_{m,M_{nb2}}(g_{m,M_{nb4}} \cdot r_{o,M_{nb4}} \cdot r_{o,M_{nb2}} || g_{m,M_{nb6}} \cdot r_{o,M_{nb6}} \cdot r_{o,M_{nb8}}) \tag{5.12a}$$

$$G_{p\ side} = g_{m,M_{pb8}}(g_{m,M_{pb4}} \cdot r_{o,M_{pb4}} \cdot r_{o,M_{pb2}} || g_{m,M_{pb6}} \cdot r_{o,M_{pb6}} \cdot r_{o,M_{pb8}}) \tag{5.12b}$$

The overall OTA gain is given as

$$G_{ota} = g_{m,M_1}(G_{n\ side} \cdot g_{m,M_4} \cdot r_{o,M_4} \cdot r_{o,M_2} || G_{p\ side} \cdot g_{m,M_6} \cdot r_{o,M_6} \cdot r_{o,M_8}) \tag{5.13}$$

The boosting circuit on the n-side is a fully differential structure. Therefore, a common mode feedback circuit is necessary. Figure 5.11 shows the circuit schematic.

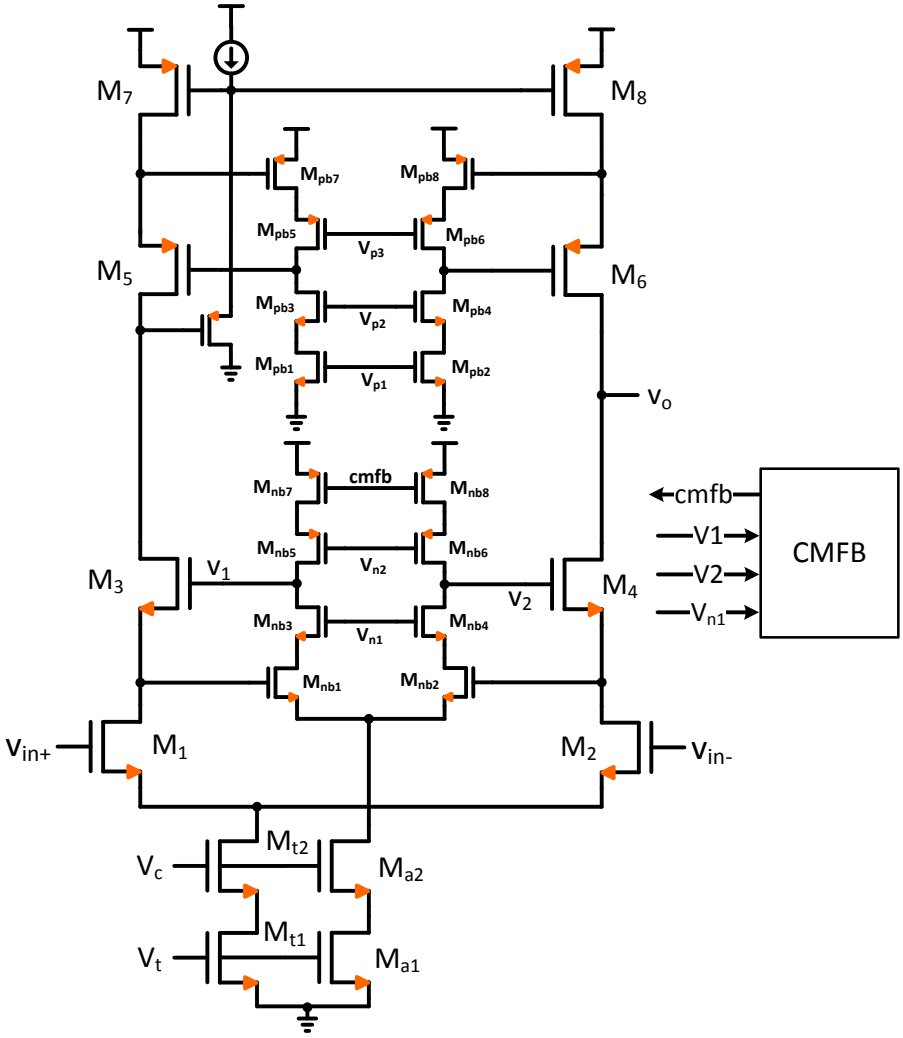


Figure 5.10 A boosted single ended telescopic OTA schematic.

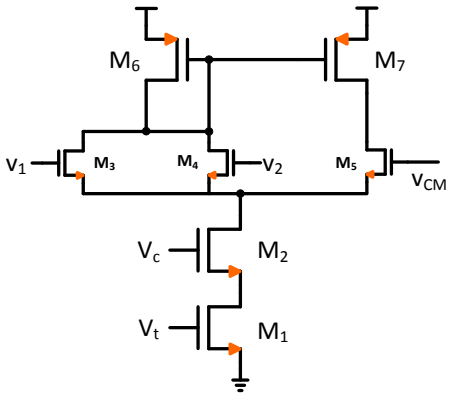


Figure 5.11 Common mode feedback circuit schematic.

CHAPTER VI

EEPROM LIFETIME AND DATA RETENTION TIME TRADEOFF MODELING

Further downscaling of CMOS process leads to the replacement of SiO_2 by HK dielectric materials and the replacement of polysilicon gate by metal gate for continuous device scaling with improved EOT. The physics of defect generation and breakdown mechanism have changed significantly with the change from SiO_2 to HK dielectrics. Understanding and modeling HK gate dielectric breakdown are still active research areas. Different schools of thought on modeling time dependent dielectric breakdown (TDDB) of the HK dielectric have been published in literature attempting to explain the breakdown mechanism. Each of them has great contribution to our understanding of the kinetics of TDDB phenomenon. This chapter first reviews frequently used TDDB models for SiO_2 gate stack. Then presents advanced TDDB model for HfO_2 gate dielectric followed by our proposed EEPROM lifetime and data retention time tradeoff model.

6.1 High-k Dielectric Gate Reliability

It is well-known that high dielectric constant films have finite number of initial traps. The CMOS fabrication process further increases initial trap density [75]. Reliability degradations such as TDDB, bias-temperature instability (BTI) and hot carrier instability (HCI) are believed to be caused by the high initial defect density [75]. The ultimate source of dielectric reliability issues is the presence of charge trapping centers or defects located in the HK film. The presence of charge

trapping centers is independent of HK film deposition technique [27], however it can be minimized during post HK deposition annealing and IL optimization [76]. Among all reliability issues, gate TDDB has been extensively studied. However, there still has not yet a well agreed universal model describing TDDB for HK dielectrics. Many research groups have their own views to the problem. Perhaps progressive breakdown is the most sound and prevailed model among others but still accompanied by opposition, which will be discussed in the following sections. TDDB of SiO_2 dielectric material has been investigated thoroughly. Several models including the E model, the 1/E model, the power law model, and the root E model have been development. However, the question that can these models be directly applied to HK dielectric materials remains. The following sections discuss frequently used TDDB models for SiO_2 dielectric and proposed models for HK dielectrics especially HfO_2.

6.2 TDDB Models for SiO_2 Gate Stack

6.2.1 E Model

The E model supports the perspective that oxygen vacancy is the intrinsic defect for breakdown and is the primary cause of TDDB. Oxygen vacancy appears because of polarized Si-O bond breakage under the influence of external applied electric field [77]. This model predicts the TDDB at low field (<10MV/cm) and high temperature that due to field enhanced thermal bond breakage [67]. The primary molecular structural unit of solid SiO_2 is the tetrahedron structure shown in figure 6.1(a). The bond strength is greatly reduced and oxygen vacancy can occur when the Si-O-Si bond angle deviates from the mean value of 150° [77]. When the Si-Si bond replaces the Si-O-Si bond, oxygen vacancy occurs as shown by figure 6.1(b).

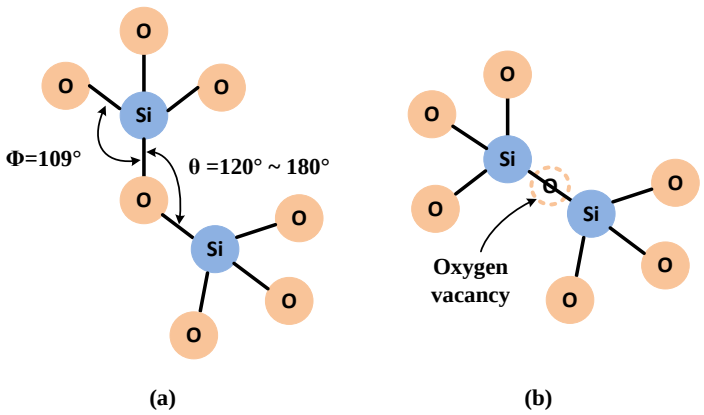


Figure 6.1. (a) Unit tetrahedron of solid form SiO_2 with bond angle between O-Si-O and Si-O-Si. (b) Si-Si bond when oxygen vacancy presents.

It is this defect in the intrinsic SiO_2 films that is considered as the cause of low-field TDDB. When Si is bonded with an unlike atom, the ionic bonding contribution to the total bonding energy increases significantly. This contribution results in polarization of the lattice [77]. In addition, an applied external electric field across SiO_2 dielectric layer shifts the electron cloud of each oxygen nucleus. This kind of distortion also induces a polarization. These two components result in a total polarizability of the lattice [77]. A carrier passing through the dielectric experiences a local net electric field which is the combination of applied external field and the polarization. *This net field can be nearly twice the applied field* [77]. Since solid SiO_2 films fracture at ~7% bond distortion [77], the high net field can easily break the bonds and results in a defect. A 10MV/cm external electric filed can cause about 2% bond distortion which induces a strong inharmonic coupling with the lattice [77]. This inharmonic coupling interacts with thermal phonons and therefore causes bond breakage after gaining enough thermal/activation energy. This is the underlying physics of the E model. Effects of the field weaken the polar molecular bond and reduce the activation energy making the bonds more susceptible to breakage [67]. As the result, time-to-failure can be expressed as [67]

$$TF = A_0 \cdot e^{-\gamma \cdot E_{ox}} \cdot e^{\frac{E_A}{K_B T}} \quad (6.1)$$

where A_0 is process/material dependent coefficient that makes TF, usually, a Weibull distribution, γ is the field acceleration parameter, E_{ox} is the electric field in the oxide, E_A is the activation energy and K_B is Boltzmann's constant ($8.62 \times 10^{-5}\ eV/K$). γ is temperature dependent and can be described as [67]

$$\gamma(T) = -\frac{\partial Ln(TF)}{\partial E} = \frac{p_{eff}}{K_B T} \quad (6.2)$$

where p_{eff} is effective dipole moment in the range of 7-14eÅ for SiO_2 and higher for higher dielectric constant materials.

6.2.2 1/E Model

The 1/E model concludes SiO_2 breakdown as a two-stage process [78]. In the first stage oxide is slowly damaged by electrical stress while the second stage is a rapid runaway process due to electrical and/or thermal runaway and leads to the formation of permanent conductive path in the dielectric [79]. The 1/E model predicts dielectric damage due to current flow through the dielectric by Fowler-Nordheim (FN) tunneling mechanism [67] [78]. During FN injection, electrons are accelerated through dielectric and cause damage to the lattice because of impact ionization. A fraction of these electrons reach the anode and excite valance band electrons to conduction band edge while left behind holes. These hot holes can tunnel back [67] [79] [80] into the dielectric causing damage due to hole induced trap generation [81] [82] [83]. This process is known as hot-hole injection. The time-to-failure is expected to have an exponential dependence on 1/E as [67],

$$TF = \tau_0(T) \cdot e^{\frac{G(T)}{E_{ox}}} \quad (6.3)$$

where $\tau_0(T)$ is a temperature dependent prefactor and $G(T)$ is a temperature dependent field acceleration parameter. $\tau_0(T)$ is given by [67]

$$\tau_0(T) = \tau_0 \cdot e^{\left[\left(\frac{-E_A}{K_B}\right)\left(\frac{1}{T}-\frac{1}{300K}\right)\right]} \tag{6.4}$$

and $G(T)$ is given by [67]

$$G(T) = G_0 \cdot \left[1 + \left(\frac{\delta}{K_B}\right)\left(\frac{1}{T} - \frac{1}{300K}\right)\right] \tag{6.5}$$

6.2.3 Power Law Voltage Model

The power law model is also known as the anode hydrogen release (AHR) model [84]. The Si-H bond at Si and SiO_2 interface can be excited by electrons and results in free hydrogen ions in the bulk of SiO_2, which can lead to defective bound generation, percolation path formation and TDDB [84]. The dependence of TDDB on voltage is given by [67]

$$TF = B_0 V^{-n} \tag{6.6}$$

where B_0 is a prefactor, exponent n is in the range of 40 to 48 for ultrathin oxide films [67]. However, the limitations of power law model include two aspects. One is that the model does not take the temperature dependence of TDDB into account and the reduction of activation energy with applied fields. The other aspect is that thicker oxide would not experience TDDB since the concentration of released hydrogen is lower compare to ultrathin oxide [84].

6.2.4 $\sqrt{E}$ Model

Current flow in high quality SiO_2 film is nearly always FN conduction and thus the damage follows 1/E model [67]. For low quality and low k dielectrics, the conduction mechanism may be Poole-Frenkel or Schottky conduction [67] [85] and results in an exponential dependence of lifetime on the square root of the applied electric field. There are three distinct root E models that

give clear mathematical expression on TDDB in low-k dielectrics [67] [85] [86]. The time-to-failure is given by equation (6.7) [67], (6.8) [86] and (6.9) [85] respectively.

$$TF = C_0(T) \cdot e^{-\alpha\sqrt{E}} \tag{6.7}$$

$$TF = AE^{-1}e^{\frac{-\beta\sqrt{E}+\varphi}{K_B T}} \tag{6.8}$$

$$TF = De^{-\alpha\sqrt{E}+\frac{\beta}{E}-\gamma} \tag{6.9}$$

Among the three models, the impact damage model, (6.9), gives a possible explanation to the damage of the dielectric. The idea is that under the applied electric filed, the kinetic energy of electrons increases and the collision of electrons with lattice atoms creates dangling bonds originated from the displacement of an atom from its normal position [85]. When a bond is broken and a lattice atom is moved, a charge trap is created. The probability of collision depends on the distance an electron can travel before being scattered and the mean free path in the dielectric. Since the collision is a momentum transfer process, whether the collision is elastic or inelastic does not matter.

All models predict the same TDDB results for electric field strength above 8 MV/cm while disparities show up at lower electric field strength [84]. Among all models, the E model gives the most conservative time-to-failure results, whereas the 1/E model gives the most optimistic results [84]. Disagreement shows up for low-field TDDB modeling in SiO_2 thin films. The E model has been successfully describing the low-field TDDB data for thick films greater than 4.0nm while in the thin oxides (< 4.0nm), the direct-tunneling current can be very high and the degradation mechanism can well be controlled by current [67]. These models are used for describing the TDDB phenomena in SiO_2 film. The direct application of these models to HK dielectric materials has been in debate. Many research groups proposed advanced TDDB models for HK dielectric

materials especially HfO_2. The following section introduces some representative school of thoughts on modeling TDDB of HfO_2 gate dielectric in the literature.

6.3 Advanced TDDB Models for HfO_2 Gate Stack

6.3.1 Progressive Breakdown

Progressive breakdown model of HfO_2 bi-layer gate stack perhaps is the most popular model that suggested by different research groups [44] [45] [46] [65] [87] [66]. Figure 6.2 shows qualitative relation between gate current and time throughout progressive breakdown process. The initial phase in breakdown process is the stress induced leakage current (SILC) phase corresponding to segment (A) in figure 6.2. During this time period, defects are generated in both IL and HK layer. Defect clusters start to form in the bulk of gate dielectric as defect sites gradually make connections. When defect clusters connect to each other, at least one pathway connects gate electrode with substrate, a percolation path is formed and the transistor is considered at the starting point of soft breakdown (SBD) indicated by point (B) in figure 6.2. Once the SBD is

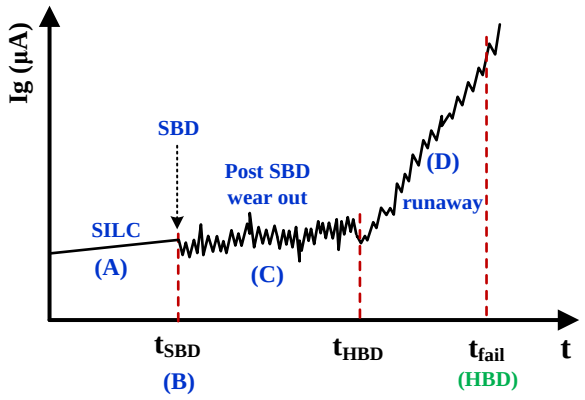


Figure 6.2 Progressive breakdown regimes. (A) Stress induced leakage current, (B) the starting point of soft breakdown (SBD), (C) post SBD wear out, (D) current runaway. [44] [45] [46]

reached, an observable sudden small increase in gate leakage current appears. However, SBD is not necessarily considered with device or circuit failure [44]. Although there is still controversy

as to which layer fails first, [88] and [89] suggest HK to be the first to fail and [90, 91, 92] suggest IL to be first, our model considers HK layer first first and uses SBD to mark the end of device lifetime to simplify the problem. A similar argument used for current runaway region (D) discussed next and used to support the selection of HK layer as the first one to breakdown. When a percolation path formed in the IL, it may not result in catastrophic failure since IL is so thin (~ 1nm) that there are not many defect sites in the close vicinity of the percolation path. It takes relatively longer time to form defect clusters in the IL. However, since the HK layer is about three to four times thicker than the IL, once a percolation path is formed in HK layer, more defect sites will be within the vicinity compare to IL. Therefore, Joule Heating has higher probability to occur near the percolation path. Once the self-enhanced process starts, the HK layer will breakdown quickly. The third stage in the progressive breakdown model is post SBD wearout or digital breakdown [45]. During this stage, random jump in the current levels corresponding to random telegraph noise is observed [45]. This is due to the stochastic capture and emission of electrons into and from the vacancy defect sites constituting the percolation path [45]. The final stage of the progressive breakdown model is thermal runaway or hard breakdown (HBD). During this stage multiple percolation paths are formed and current increases significantly. HBD stage also involves migration of the metal atoms/ions from the gate into the dielectric and protruding all the way to the substrates [45]. Device completely fails once the HBD is reached. The following effort discusses each stage in more details.

(A) SILC

When a HK gate stack transistor is stressed by voltage and/or temperature, defects are randomly generated in the bulk of dielectrics. Excess current is induced and causes deviation of IV characteristic from theoretical tunneling mechanism. This phenomenon corresponds to the SILC stage. During this stage, gate current follows a power law relation with time and the time exponent is in the range from 0.3 to 0.7 [47] [93]. As more and more defects are generated, more

electrons are able to reach the gate by means of trap-assisted tunneling (TAT) [94] [95]. As defects are gradually generated, defect clusters form in the bulk of dielectric which reduces the average tunneling distance of electrons (when considering NMOS at inversion). This increases tunneling probability and results in current flow. The tunneling probability is exponentially dependent on the barrier height and governed by the Wentzel-Kramers-Brillouin (WKB) approximation [96]. Cartier and Kerber suggest that SILC is a recoverable phenomenon with negative gate voltage applied to the HK gate stack [62].

(B) SBD

As SILC stage continues, more defects are generated and clusters of defects start to form. At the moment one particular combination of clusters connects the gate and the substrate, a percolation path appears [97]. For dielectric thickness larger than about 4 to 5nm, TDDB is the end point of oxide lifetime due to fast transition to thermal runaway. The percolation results in nonreversible damage to the gate dielectric leading to device malfunction [45]. For thin dielectric films of thickness is less than 2 to 3nm, the breakdown process is soft, which means the catastrophic failure does not appear abruptly but undergoes a wearout period [45]. The reason for this difference is that defect density is much higher in the thicker dielectric and many defects surround the percolation path eases the occurrence of Joule Heating process [45]. This can easily result in positive feedback on the wearout process that aggravates further defect generation until the dielectric breaks down. For thinner oxide, few defects present within the vicinity of the percolation path and alleviate the breakdown process [45].

(C) Post SBD Wearout

Continuing stress on the dielectric post percolation path has formation results in observable random gate current jump corresponding to random telegraph noise [45]. The current jump is due to carriers capture and emission into and from the vacancies. This is also known as the band to

band tunneling. According to some researchers, post SBD wearout is a safe region to continue operate the transistor but will enter breakdown gradually [45].

(D) Current Runaway

Further defect generation in the post SBD wearout process, especially close to the percolation path, leads to increase of local temperature due to Joule Heating. The rise in temperature enhances subsequent defect generation which again increases temperature and current density. The positive feedback mechanism will dilate percolation path and thin down the oxide vertically [45]. Eventually the dielectric loses its insulating property and becomes conductive indicating the end of lifetime.

6.3.2 Generated Subordinate Carrier Injection Model

K. Okada et al. [89] proposed a Generated Subordinate Carrier Injection Model (GSCI), which attributes dielectric degradation and breakdown to the subordinate carriers. The argument is that the subordinate carrier charge is constant with respect to gate stress voltage. Thus, the degradation and breakdown are caused by subordinate carriers. However, with one exception that electron charge density varies with low gate voltage values for $HfALO_X$ samples. At higher stress voltages, the electron charge density is constant. Authors consider the cause of the exception is the presence of a trap-assisted conduction current component, I_{TA}, in the subordinate carrier current. Subordinate carrier current consists of two components, one is the trap-assisted conduction component, and the other is dominating carrier component, I_{DC}, responsible for dielectric degradation and breakdown [89]. Authors presume the I_{TA} component does not contribute to the defect generation and the breakdown, only the I_{DC} does. The GSCI model can be summarized into three aspects: 1) the subordinate carrier controls the degradation and breakdown of the device, 2) breakdown occurs when injected dominating carriers reach the breakdown threshold, 3) both electrons and holes can be the dominating carrier [89].

6.3.3 A Percolation Model with Different Defect Generation Rates

Advanced Micro Devices (AMD) group, T. Nigam [70], made observations that HK dielectric gate stacks are characterized by short breakdown times and shallow Weibull slopes. These observations are explained by a percolation model with different defect generation rates in the HK layer and IL. The difference in defect generation rate results a bimodal distribution with a transition from a shallow to steep Weibull slope [70]. Test device area range is from 0.006 um^2 to 1000 um^2 and stressed at 125°C. Two different IL thickness were investigated (paper does not specify values) with 2nm thick HK layer. Results shows that for both NMOS and PMOS, larger area devices have larger Weibull slopes. NMOS devices show significant current increase prior to breakdown which is due to either a progressive component or SILC. Using an AC stress where devices are allowed to discharge at a fixed negative gate voltage, no significant increase in gate current is observed. Therefore, authors conclude the absence of progressive component. And the increase of gate current is attributed to SILC and defects generated in the HK layer. Similar results apply to PMOS as well. The progressive component may be used for explaining the transition of Weibull slope [70]. Since this work shows no progressive component, different defect generation rates in the IL and HK layer are used to explain the transition. Three dimensional kinetic Monte Carlo simulations are performed. Observations made from simulation results are (1) an increasing defect generation rate in the HK layer decreasing time to breakdown, and (2) TDDB distributions are bimodal for non-uniform defect generation in the HK layer and the IL. Transition occurs at lower Weibits since the defect generation rate of the HK layer is higher than the IL. Area effect on TDDB can be summarized as (1) for small areas, defect density in the HK layer is high and the time to breakdown and Weibull slope are limited by the IL, (2) for large areas, Weibull slope is determined by the complete stack thickness.

6.3.4 An All-in-one TDDB Reliability Model

Researchers T. Kauerauf, R. Degraeve et al. [98] [99] have presented an all-in-one TDDB reliability predicting model. Authors suggest that the constant voltage stress (CVS) extrapolation of SBD is challenged because (1) due to low breakdown voltages and low Weibull slope, the extrapolated SBD maximum applicable voltage for 10 years is significantly below the operating voltage, (2) high intrinsic gate leakage current comparing to the percolation current makes the SBD and its trigger moment hard to observe. In order to overcome the limitations, a combined TDDB extrapolation including both SBD and HBD is presented. Lifetime is no longer plotted as a function of the applied gate voltage but as dielectric area vs. gate voltage plot including three regions: SBD free region, region of multiple SBD (wearout) and HBD region, shown by author's figure 6.3. It predicts no SBD on 0.1cm^2 device after 10 years if V_G<0.52V.

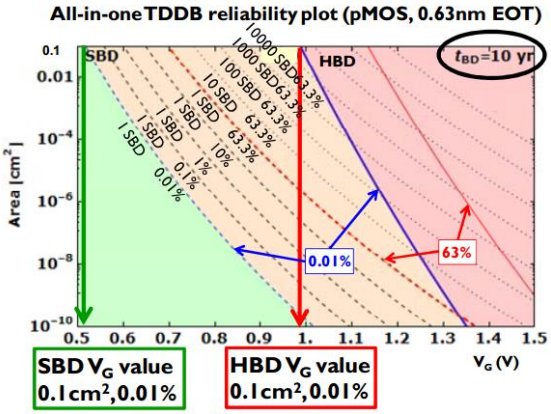


Figure 6.3. All-in-one TDDB reliability for a 0.63nm EOT PMOS. Adapted from source [98].

If V_G=0.9V more than 1000 SBDs will be created after 10 years and for V_G=1V, 0.01% of the chips will fail due to HBD. Figure 6.3 is generated using reliability parameters from the SBD and wearout distributions: the Weibull slopes β_{SBD} and β_{WO}, the 63% values η_{SBD} and η_{WO} and voltage acceleration power-law exponents nSBD and nOW. There relation to the HBD is given as [98] [99]

$$F_{HBD}[t] = 1 - \exp\left[-\left(\frac{\eta_{WO}}{\eta_{SBD}}\right)^{\beta_{SBD}} \cdot \int_0^{\left(\frac{t}{\eta_{WO}}\right)^{\beta_{WO}}} \left(\frac{t}{\eta_{WO}} - u^{\frac{1}{\beta_{SBD}}}\right)^{\beta_{SBD}} \cdot e^{-u} du\right] \quad (6.10)$$

6.4 Proposed Comprehensive Model

6.4.1 Model Overview

This model is expected to project long term EEPROM lifetime and data retention time tradeoff with respect to different programming voltages in order to establish optimal programming protocols for the trap charge based EEPROM. The proposed comprehensive model block diagram is shown in figure 6.4. The key parameters that connect the EEPROM data retention time and lifetime are programed threshold shift, $\Delta V_{th,programed}$, programming electric field, $\xi_{programming}$, across the gate oxide indicated as orange circle v_1 and v_2 respectively in figure 6.4. Block 1 relates the desired EEPROM data retention time to the programed threshold shift $\Delta V_{th,programed}$. Data retention requirement or stored charge loss is raised from trapped carriers gradually detrapped from the gate dielectric because of thermal agitation and thermally activated tunneling [100]. After the programed threshold shift is obtained, current transport mechanisms of carriers tunneling through HK-IL-Si barriers are used to find corresponding programming electric field, $\xi_{programming}$. Different charge transport mechanisms are discussed in section 3.3. Fowler-Nordheim tunneling is used in our model. This part of the model is depicted by block 2 in figure 6.4. Block 3 is considered as the heart of our comprehensive model. It provides lifetime with respect to different programming electric field strength. It has been discussed in section 6.3 that there are many different proposed HK dielectric device lifetime models in the literature. In order to avoid any bias of different models, we use 3D kinetic Monte Carlo (kMC) method to extrapolate the lifetime of the device. The only potentially controversial point is the defect generation rate used in the 3D kMC. Different rate equations are given by different research groups [70] [71] [72]. We have selected the one we considered the most consistent with other literatures based on our understanding. The

three blocks constitutes the comprehensive lifetime - data retention time tradeoff model. In addition to the model itself, we suggest that the average threshold of each EEPROM die should be measured to improve device lifetime. Since transistor threshold voltage varies due to technology process and mismatch and the programming relies on threshold voltage modification, the threshold variation has to be taken into account for programming. For this reason, we introduce block 4 that is able to measure the average threshold voltage of each EEPROM die at the beginning of its operation. Detailed implementation of each bock is introduced in the following sub-sections.

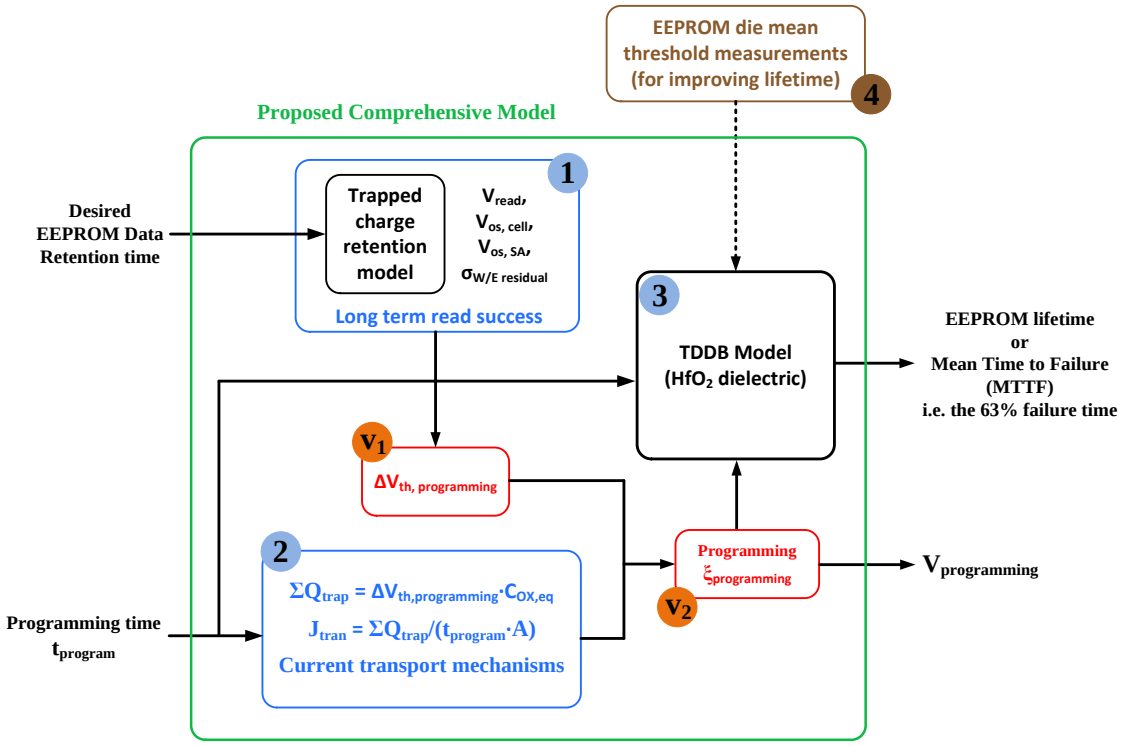


Figure 6.4 Proposed trap charge based EEPROM lifetime - data retention time tradeoff model flow diagram.

6.4.2 Block One

Block 1 in figure 6.4 serves the purpose of relating desired EEPROM data retention time to the required programed threshold shift, $\Delta V_{th,programed}$, needed to achieve the desired data retention

time. The desired programed threshold shift is determined by the long term charge retention loss, required read voltage and input offset voltage of the storage cells and sense amplifier as shown by equation (6.11)

$$\Delta V_{th,programed} \geq V_{read} + \sqrt{V_{os,cell}^2 + V_{os,SA}^2 + \sigma_{W/E,residual}^2} + \Delta V_{th,loss} \qquad (6.11)$$

where $V_{th,programed}$ is the programmed threshold shift, V_{read} is the input voltage the sense amplifier to achieve the desired read time, $V_{os,cell}$ and $V_{os,SA}$ are the offset voltage of the storage cell and sense amplifier respectively, $V_{W/E,residual}$ is write/erase residual variation and $\Delta V_{th,loss}$ is undesired threshold shift due to trapped charge loss over time. From [23] and [56], the residual variation is small compare to the programming threshold shift and has negligible effect on memory operation. The desired read voltage is derived from [101]

$$t \approx \frac{1}{\omega_t} \ln\left(\frac{\Delta V_{final}}{\Delta V_{read}}\right) \qquad (6.12)$$

Offset voltages of the storage cell and sense amplifier are given by [102]

$$V_{os,cell} = \frac{A_{VT,n}}{\sqrt{WL}} \qquad (6.13a)$$

and

$$V_{os,SA} = \frac{A_{VT,p}}{\sqrt{WL}} \qquad (6.13b)$$

respectively, where $A_{VT,n}$ and $A_{VT,p}$ are Pelgrom coefficients for NMOS and PMOS transistors respectively. The 32nm SOI CMOS process has $A_{VT,n}$ approximately 2.3 mV·μm and $A_{VT,p}$ approximately 3.4 mV·μm. Offset voltage can be reduced by increasing transistor area according to equation (6.12). $\Delta V_{th,loss}$ in equation (6.11) is threshold shift due to trapped charge loss from the HK dielectric layer after several years. It is the characterization of charge retention. Ten years

is selected as a standard retention criterion for nonvolatile memory applications. Retention characteristics are important to nonvolatile memory devices. The charge loss is determined by tunneling leakage under weak fields. Possible causes of charge loss include defects in the tunnel oxide, i.e. the interfacial layer (of minimal concern), defects in the blocking layer, mobile ion contamination, and detrapping through charge trapping layer surrounding insulation [103]. The experiment conducted for studying data retention is the Arrhenius test. Arrhenius testing is an accelerated test method completed at elevated temperature to accelerate device aging and extract lifetime or other characteristics under normal operating temperature for device. To understand the retention characteristics of HK nonvolatile memory, flatband voltage shift as a function of time for various baking temperatures (85 - 225°C) is measured. Certain carriers captured by the traps close to SiO_2/HfO_2 interface in the HfO_2 layer have shorter capture times. Carriers captured by deep traps whose energy levels are within the HfO_2 bandgap are desired for memory applications as will be discussed in section 6.4.4. Table 6.1 summarizes several retention results for $HfO_2/SiO_2/Si$ gate stack from selected publications. Reference [56] uses the fabrication process that identical to our work and reported 16% V_{th} loss, which will be used to determine the required programmed threshold shift.

Table 6.1. Retention characteristics of $HfO_2/SiO_2/Si$ gate stack with different layer thickness

Reference	Process	IL layer material and thickness	HK layer material and thickness	Stress conditions	Projection of 10 years retention time
[104]	160nm SOI-NMOS	ONO (oxide-nitride-oxide), 2/2/2 nm	HfO_2, 3nm	$V_{g,\ program}$ = 16V $t_{program}$ = 2.5ms	estimate 14% charge loss
[56]	32/22nm SOI	SiO_2, 1nm	HfO_2, 3nm	$V_{g,\ program}$ = 2V $t_{program}$ = 10ms	16% V_{th} loss, baked at 85°C
[105]	-	SiO_2, 3nm	HfO_2, 8 - 2nm	$V_{g,\ program}$ = 18V $t_{program}$ = 1s	30.8% V_{th} loss
[106]	-	SiO_2, 5nm	HfO_2, 25nm	$V_{g,\ program}$ = 13V $t_{program}$ = 1s	~34% charge loss, baked at 85°C
[107]	-	SiO_2, 4.5nm	HfO_2, 6nm	$V_{g,\ program}$ = 18V	~44% V_{th} loss, baked at 200°C

6.4.3 Block Two

With required threshold shift found from equation (6.11), the total amount of trapped charge can be calculated using relation

$$\Sigma Q_{trap} = \Delta V_{th,programed} \cdot C_{ox,eq} \tag{6.14}$$

where $C_{ox,eq}$ is the equivalent gate oxide capacitance. The current density due to these transported charges can be calculated using

$$J_{tran} = \frac{\Sigma Q_{trap}}{t_{program} \cdot A} \tag{6.15}$$

where $t_{program}$ is the time duration of programming gate applied, A is gate oxide area. Using Fowler-Nordheim tunneling mechanism, the electric field strength across the gate oxide corresponding to the transport current density can be found by [49] [73]

$$J_{tran} = J_{FN} = A_{FN}\xi^2_{\text{programming}} e^{-\frac{B_{FN}}{\xi\text{programming}}} \tag{6.16}$$

where A_{FN} and B_{FN} are given as

$$A_{FN} = \frac{q^2}{8\pi h\phi_0} \tag{6.17}$$

and

$$B_{FN} = -\frac{4}{3}\sqrt{\frac{8\pi^2 m}{h^2}}\frac{(q\phi_0)^{3/2}}{q} \tag{6.18}$$

In equation (6.17) and (6.18), m is electron mass, ϕ_0 is barrier height of oxide-silicon interface, h is Planck's constant. This electric filed is the required programming electric field strength corresponding to the desired data retention time stated in block one. This electric field strength is used in block three for lifetime extraction.

6.4.4 Block Three

As discussed in Chapter 3, oxygen vacancy creation, transportation, and extinction are relevant to the degradation of HK gate dielectric and the lifetime of nonvolatile memory which using the HK device. Many models have been introduced in literature. For our model development, we have decided to use three dimensional kinetic Monte Carlo (3D kMC) algorithm to extrapolate the dielectric lifetime for a required programming electric field. The general idea is first dividing the bulk dielectric material into small cubes in a 3D grid according to the dielectric geometry and defect size. Each cube represents a lattice site that has the potential of turning into a defect by the programming electric field. Then, using a random generator to randomly generate defects (a marked or occupied lattice location) in the 3D lattice grid according to a generation rate. Once a defect appears at a lattice location its surrounding lattice locations will have higher chance to become the next defect location because of higher local electric field strength. The generation rate at the surrounding locations will then be updated. When one or more neighboring locations become defective, a defect cluster forms. All touching clusters have the same cluster label. When

a new defect location joints two un-touching clusters, they all form a new cluster and chose the lowest-numbered label of the cluster as this new cluster label. When the same cluster label appears in both the top layer and bottom layer of the lattice, there is a percolation path connecting the top and bottom layers. This percolation path marks the SBD stage of the dielectric as discussed in section 6.3.1. This process of checking cluster formation and percolation path is known as the Hoshen-Kopelman algorithm [108] which is based on union-finding algorithm [109]. For the purpose of this work, simulation terminates when a percolation path formed and we consider the soft breakdown as the end of device lifetime. The key parameter for the kMC simulation is the defect generation rate. It is mainly a function of applied electric field and is given by [110] [111]

$$r_{defect} = \nu \cdot e^{-\frac{E_a - p_{eff} \cdot \frac{2+\varepsilon}{3} \xi_{\text{programming}}}{K_B T} \cdot t} \tag{6.19}$$

where ν is lattice vibration frequency, E_a is the activation energy, p_{eff} is effective dipole moment, ε is dielectric constant and t is time elapsed. The activation energy is defined as the energy difference between the trap site and the conduction band of Si-substrate [112]. Defect levels in the HfO_2 HK dielectric layer can be classified into three groups according to their locations within the bandgap relative to the Si-substrate conduction band and valence band [112]. Figure 6.5 shows the simplified energy band diagram of group allocation. Trap levels in group 1 are considered as shallow levels where trapped carriers can easily de-trap after programming voltage is removed. This group of traps does not provide long term memory functionality. Group 2 levels whose energy span maps to the Si-substrate bandgap are the desired trap energy states. Under non-programming condition, carriers trapped in group 2 energy levels detrap slowly to the Si-substrate (if NMOS is considered) based on their physical distance from the substrate, the activation energy and the quality of the dielectric material. Threshold shifts due to carriers trapped in there levels are suitable for memory application. Group 3 trap levels shown in figure

6.5 are levels below the valance band of Si-substrate. Carriers trapped in there levels give rise to the fixed oxide charges [112] and are difficult to remove under normal erase voltages. The desired trap energy is about 1.2eV to 2.3eV below the conduction band edge of the HfO_2 dielectric as shown in figure 6.5.

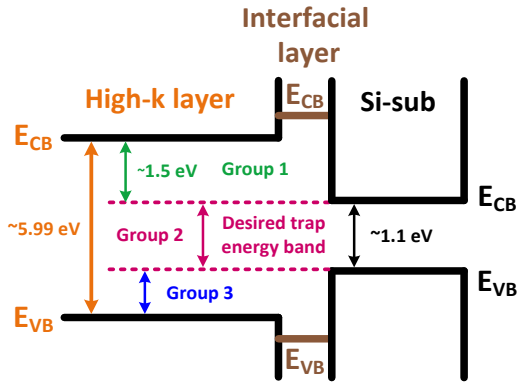


Figure 6.5. Different groups of defect levels within the bandgap of HK dielectric layer [112].

After the defect generation rate is determined, 3D kMC algorism can be applied to extrapolate the lifetime of the device. In 3D kMC simulation, dielectric material is divided into a volume of small cubes shown by figure 6.6 inset. Each cube represents a lattice site that has the potential to become a defect under the stress of programming electric field. Number of layers assigned to HK layer and IL are determined by the physical thickness and defect size. The physical thickness of HK layer and IL layer used is 4nm and 1nm respectively. Defect size is about 0.6nm [71] [70]. Thus, there are 6 layers used in our simulation. Each layer is a 30 x 30 cube array. For easing illustration purpose, figure 6.6 shows a two dimensional kMC percolation diagram. Our modeling

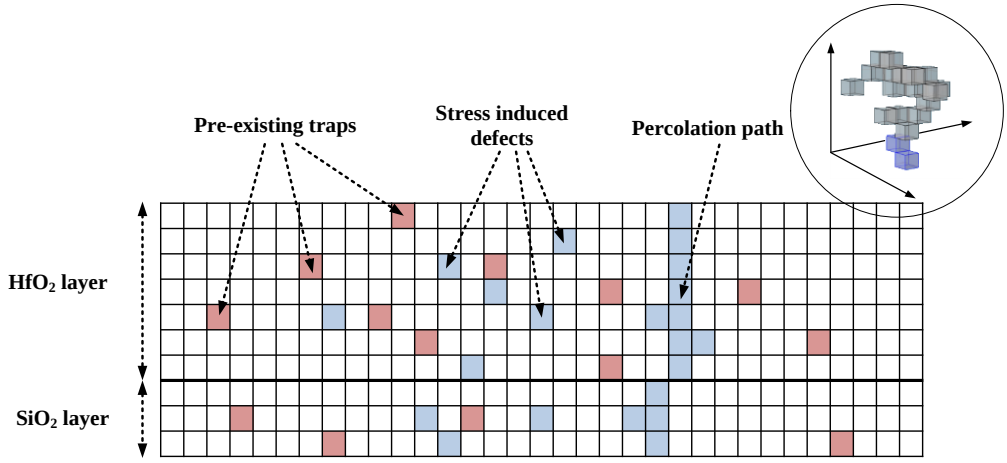


Figure 6.6 2D percolation diagram showing SiO_2 interfacial layer and HfO_2 HK layer.

procedure uses the basic 3D kMC algorithm [113] with some modifications and summarized as the following:

1. Initialize parameters and set time to zero

2. Assign initial defect generation rate r_i to each lattice site.

3. Calculate the cumulative partial sum of each defect generation rate associated with a lattice site, $r_{par,j} = \sum_{j=1}^{N} r_i$ for i = 1, …, N where N is the total number of lattice sites or the cubes.

4. Generate a uniform random number, u_1, between 0 and 1.

5. Mark the first lattice site that satisfies $r_{par,j} > u_1 \cdot r_{total}$, where r_{total} is the sum of all site generation rates.

6. Generate a new uniform random number, u_2, between 0 and 1.

7. Calculate time elapse $t = t + \left(-\frac{logu_2}{r_{total}}\right)$.

8. Update defect generation rate at all defect locations according to equation (6.19).

9. Check for percolation path.

10. If no percolation path has formed, return to step 3. Otherwise simulation ends.

This procedure is simplified to a program flow chart shown by figure 6.7. The simulated lifetime or mean time to failure is plotted on a Weibull plot. MATLAB code for the model implementation is in the appendix part II.

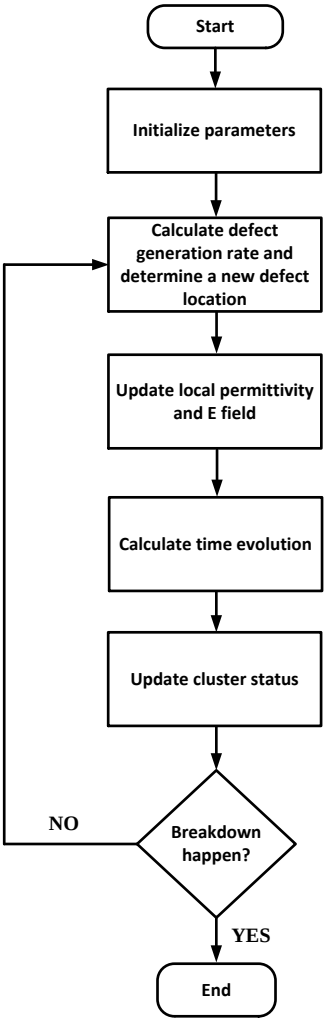


Figure 6.7 3D kinetic Monte Carlo simulation program flow chart.

6.4.5 Block Four

As mentioned above, the knowledge of average threshold voltage of a die can improve EEPROM lifetime and yield when different dies from a same wafer or from different wafers are considered. The remaining of this section is dedicated to explain the proposed method for threshold measurement. To illustrate the idea, we use the 1/E model. We rewrite the 1/E time-to-failure equation in terms of programming voltage as

$$T_{BD} \cong \tau_0 e^{\frac{K\, t_{ox}}{V_g - V_{th}}} \tag{6.20}$$

where K is associated with the carrier tunneling and τ_0 is a temperature dependent pre-factor [84]. This equation reveals an exponential relationship between the transistor lifetime and gate overdrive voltage, $V_g - V_{th}$. Therefore, accurately controlling the gate voltage and knowing the threshold variation are critical for lifetime improvement. There are two sources of contribution to threshold variation; one is transistor variability mainly due to channel dopant fluctuation and line edge roughness, the other is threshold variation due to fabrication process gradient. Transistor variability dominates when considering silicon area less than about 200μm diameter. Threshold variation due to process gradient dominates for area larger than about 200μm diameter. In this application, the result of transistor variability is threshold mismatch of the differential storage cell. Our four EEPROM banks are laid out in an area about 260μm x 268μm. Therefore, threshold mismatch dominates over the EEPROM banks. The threshold mismatch is given by (6.13a). As shown by equation (6.13a), threshold mismatch between each cell is reduced by designing the cell with sufficient area. As shown by equation (6.11), the threshold mismatch needs to be less than the write/erase residual so that the write/erase residual is the dominant error contribution comparing to threshold mismatch. In addition to the threshold mismatch, threshold variation due to process gradient is considered for lifetime improvement. When dealing with threshold variation due to process, the four banks are seen as a single entity and the average

threshold voltage across it is measured. The average threshold of each EEPROM die (four banks) from different region of a wafer or from different wafers is different because of process variation. Thus, the ability of measuring the average threshold for each die can reduce lifetime degradation shown by equation (6.20). This means the programming voltage can be adjusted for targeted programming threshold shift according to the measured average native threshold. The choice of a universal programming voltage can be avoided to improve memory lifetime.

We proposed a simple method that enables the average threshold measurement across four banks. Figure 6.8 shows the conceptual diagram. Four test transistors are placed at the four corners of the full EEPROM bank. These test transistors have a sufficient number of fingers that the threshold mismatch between them is negligible comparing to threshold variation due to process while satisfying silicon area constraint. The four test transistors are connected in parallel and their average threshold is measured using circuit shown in figure 6.8(b). The reference current in figure 6.8(b) is the current at which the threshold is measured. The top current mirror output is scaled down four times to average the current sum of the four test transistors. As the DAC sweeps the gate voltage of the test transistors, their average current will be compared with the reference current. The result is converted to logic level by a comparator. Once knowing this mean threshold for each die, the programming voltage can be adjusted accordingly such that the low threshold dies will not be stressed as hard as high threshold dies. Lifetime is therefore prolonged. This section gives the overview of proposed MTBF model and an average threshold measurement method for a die. Next section shows the LDO implementation for generating the programming voltages.

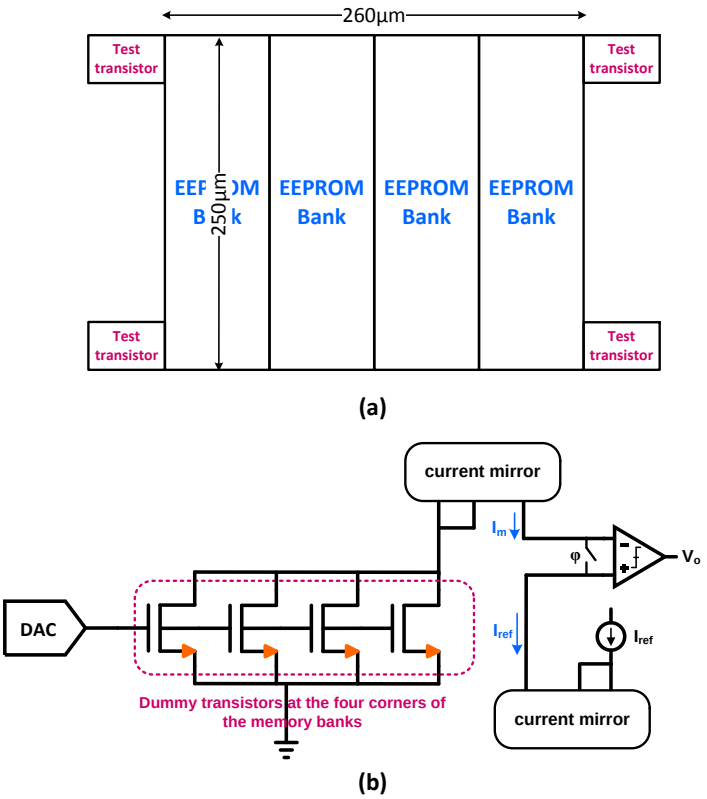


Figure 6.8 (a) Four test transistors are placed at four corners of the full EEPROM bank for Vth measurements. (b) Proposed threshold voltage measurement circuits.

CHAPTER VII

RESULTS

This chapter presents the results of this work. These results consist of three parts. The first part is CPU/EEPROM interface functional simulation waveforms. These waveforms capture the interface and EEPROM activity to show the functional correctness of memory operations and the interface logic. The second part is LDO simulation. This part includes LDO closed-loop AC simulation waveform and output transient simulation waveforms. The third part is the HfO_2 gate stack lifetime extrapolation using 3D kMC algorithm.

(Run directory: ~/VHDL/EEPROM_2018/top/top.xise)

(Run TopTB-behavior (TopTB.vhd))

(Simulate Behavior Model)

7.1 CPU/EEPROM Interface Simulation

This section presents the behavioral simulation of the CPU/EEPROM interface. The behavioral activity is depicted by memory state transitions among different operations. The BUSY signal is used to verify the CUI operation for CPU and EEPROM exchange control. Figure 7.1 shows the interface behavior for a block write operation. The operation begins with CPU load the initialization, command, and data registers. Since there are 16 words (i.e. a block) to be written as

an example, it takes 16 cycles for CPU load the data register. This results the staircase like waveform in the figure. After the data register is loaded, control is passed from CPU to EEPROM and write operation starts. The BUSY signal is set indicating EEPROM has control to CUI and its operation starts.

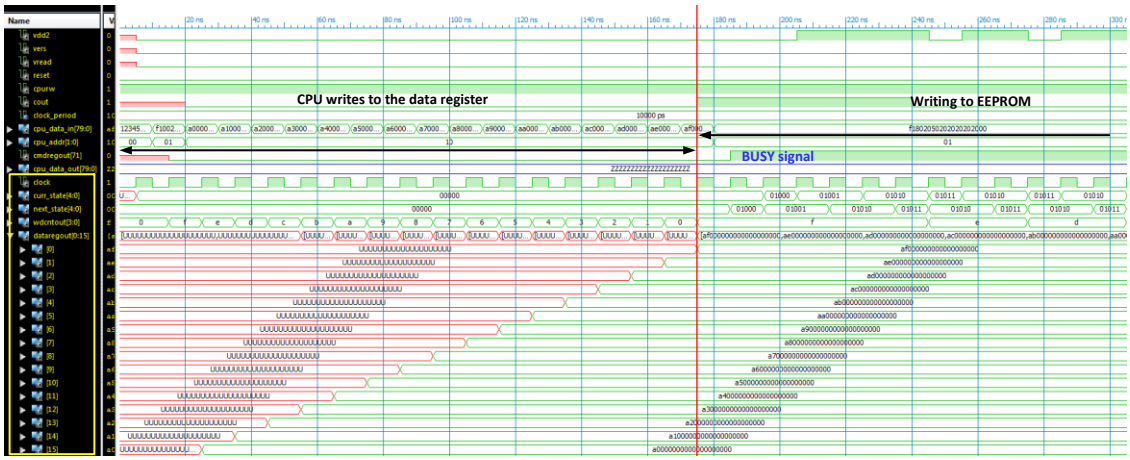


Figure 7.1 EEPROM write operation. Control transition from CPU to EEPROM.

It takes 16 cycles for the EEPROM to write the block of word. To avoid redundancy, not all 16 cycles are shown in the captured waveform but only the moment of memory making state transitions. Figure 7.2 shows the memory operation transition from write to read. As discussed in Chapter 4, in order to enable CPU to verify data correctness, read operation stats automatically after write. This transition is indicated by the red line in figure 7.2.

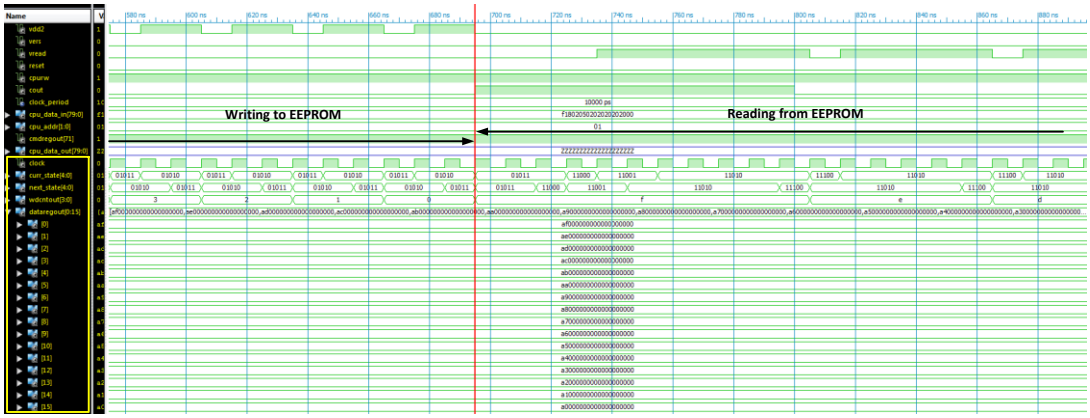


Figure 7.2 EEPROM write operation. Operation transition from write to read.

After read operation is complete, EEPROM return CUI control to CPU by reset the BUSY signal as indicated in figure 7.3. After BUSY signal is reset, CPU can access data register to recover the data just written to the EEPROM and EEPROM enters idle state. CPU can also discard the information in the data register by overwrite it with new value.

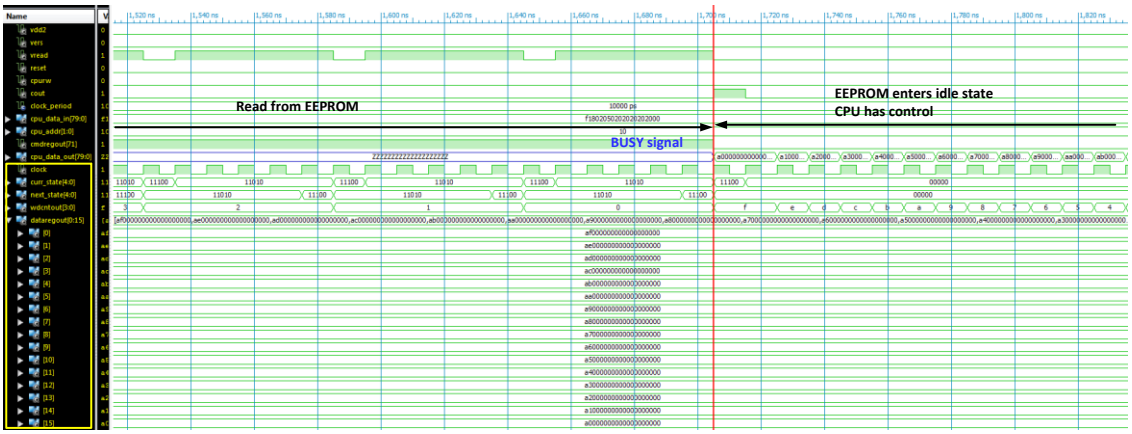


Figure 7.3 EEPROM write operation. Control transition from EEPROM to CPU.

Figure 7.4 shows erase operation waveform. After BUSY signal is set, control is passed from CPU to EEPROM and erase operation starts. A block of word is erased in one operation. The waveform in figure 7.4 shows the first block of words of bank0 is erased. After erase operation, read operation starts. The transition from erase to read is marked in figure 7.4.

Figure 7.4 EEPROM erase operation. Control transition from CPU to EEPROM. Operation transition from erase to read.

After read operation is complete, data is written to the data register for CPU to recover. This part of process is identical to the write operation. When read is complete, BUSY signal is reset, CPU regain control from EEPROM. EEPROM enters idle state. This is shown in figure 7.5.

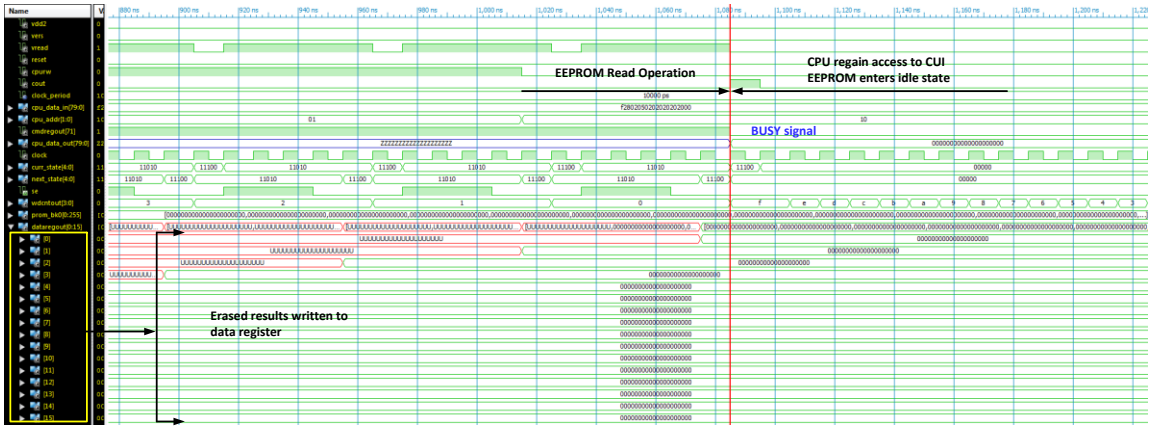


Figure 7.5 EEPROM erase operation. Control transition from EEPROM to CPU.

For the completeness purpose, read operation alone is shown in figure 7.6 and 7.7. The operation is similar to write and erase. To avoid unnecessary detailed description, only the transition moment of read operation are presented for showing the interface correct functionality.

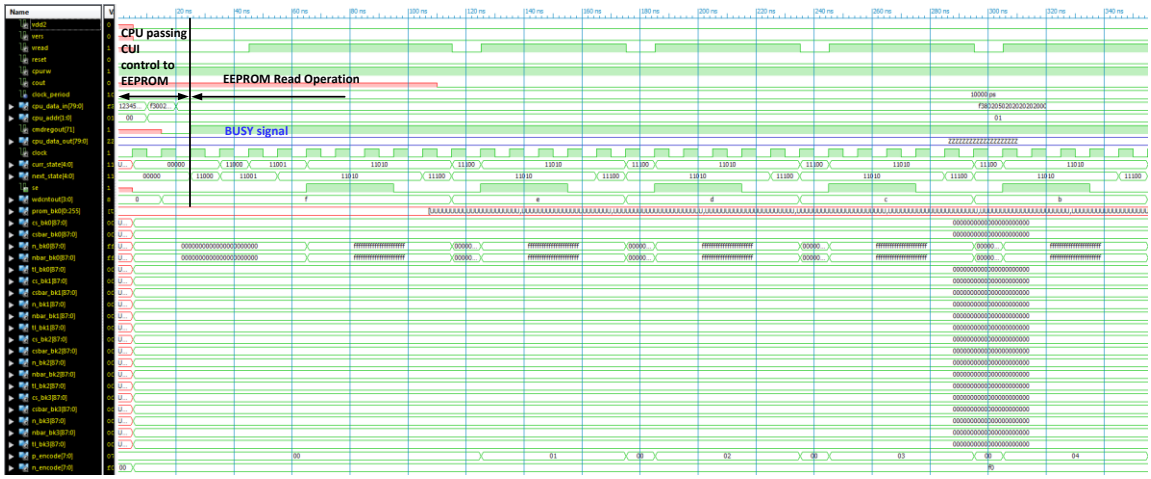


Figure 7.6 EEPROM read operation. Control transition from CPU to EEPROM.

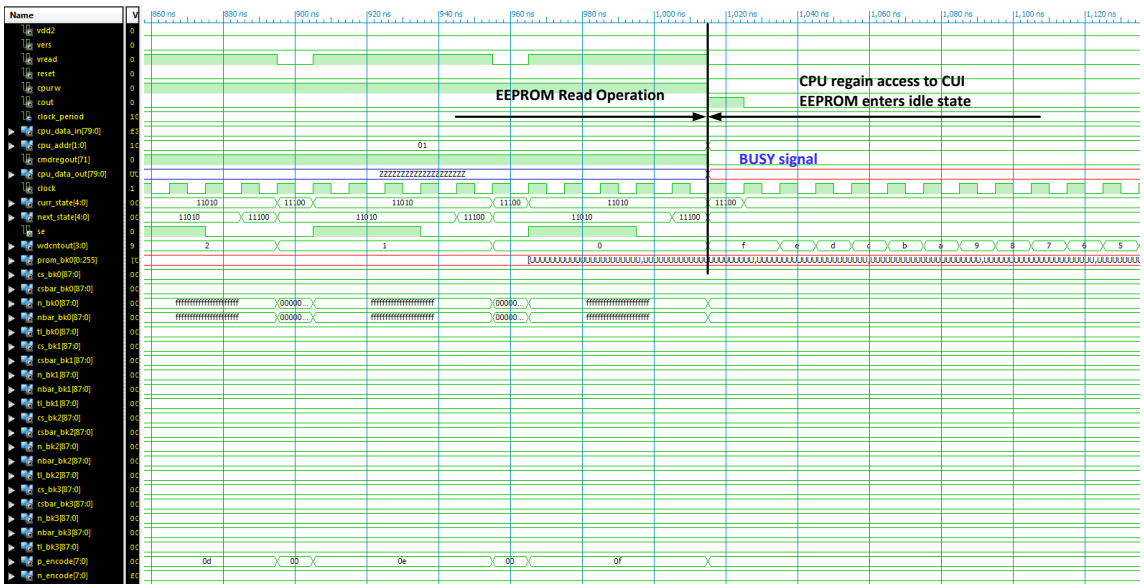


Figure 7.7 EEPROM read operation. Control transition from EEPROM to CPU.

7.2 LDO Simulation

Figure 7.8 shows the simulation schematic of the LDO. The main circuit block includes start up circuit, bandgap voltage reference, bias circuit, single-ended boosted telescopic OTA, common mode feedback circuit, second stage pass transistor, and feedback resistor network. The circuit is simulated at typical-typical process corner. Figure 7.9 shows closed-loop Bode magnitude and phase plots. The closed-loop gain, i.e. LDO gain, is 1.5. Closed-loop GBP is 638.68kHz. Phase margin is 131°.

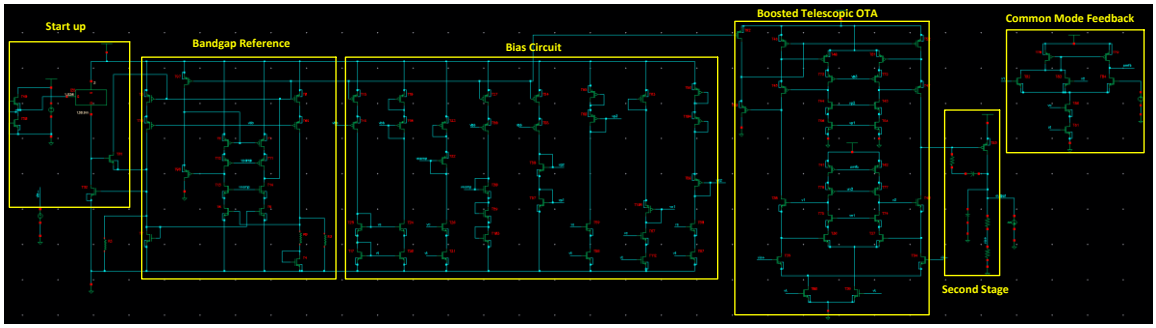


Figure 7.8 LDO simulation schematic.

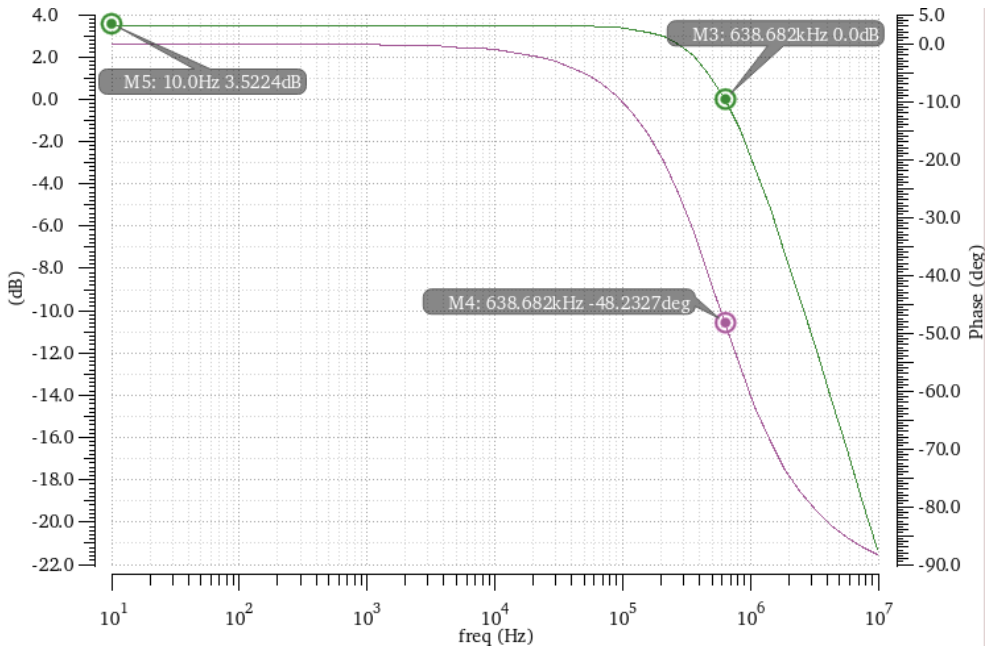


Figure 7.9 LDO closed-loop Bode magnitude and phase plots.

Figure 7.10 and figure 7.11 show the transient response of the LDO for 0.6V output and 2.5V output respectively. The load current pulse is 2mA with edge time of 10µs. Its period is 20ms with 50% duty cycle. For 0.6V output, LDO has 2.5% overshoot on the falling load current transition and 2% overshoot on the raising load current transition. The LDO settling time of falling and raising load current edge is 12µs and 11µs respectively. For 2.5V output, LDO has 0.13% overshoot on the falling load current transition and 0.14% overshoot on the raising load current transition. The LDO settling time of falling and raising load current edge is 12µs.

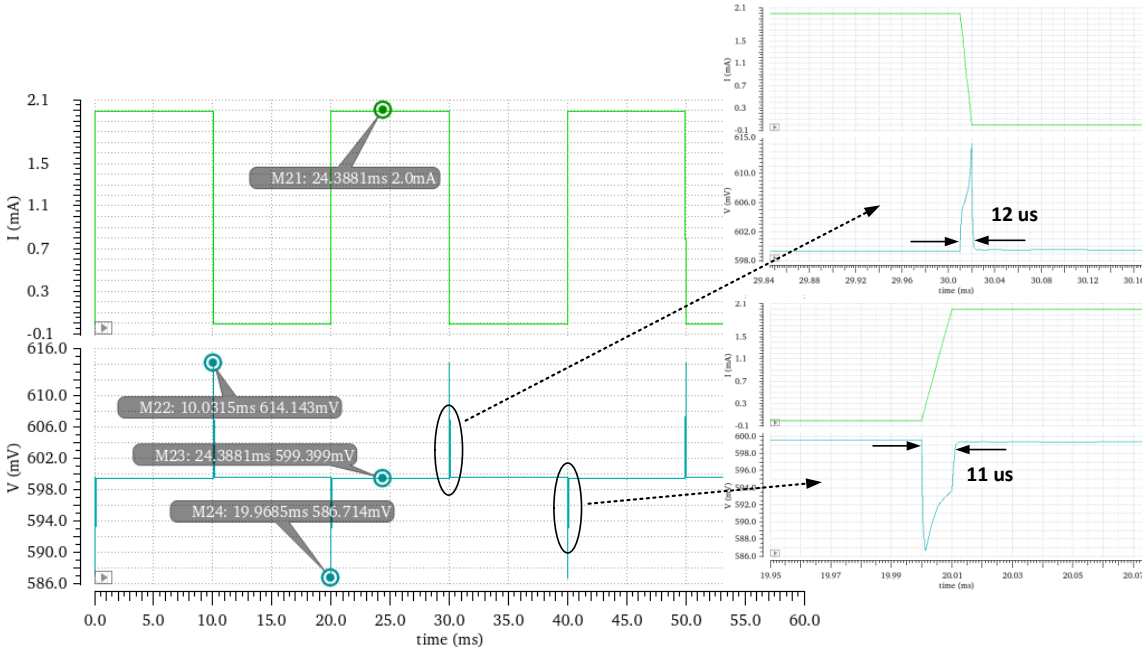


Figure 7.10 Output transient of the LDO with 0.6V output and 2mA load current.

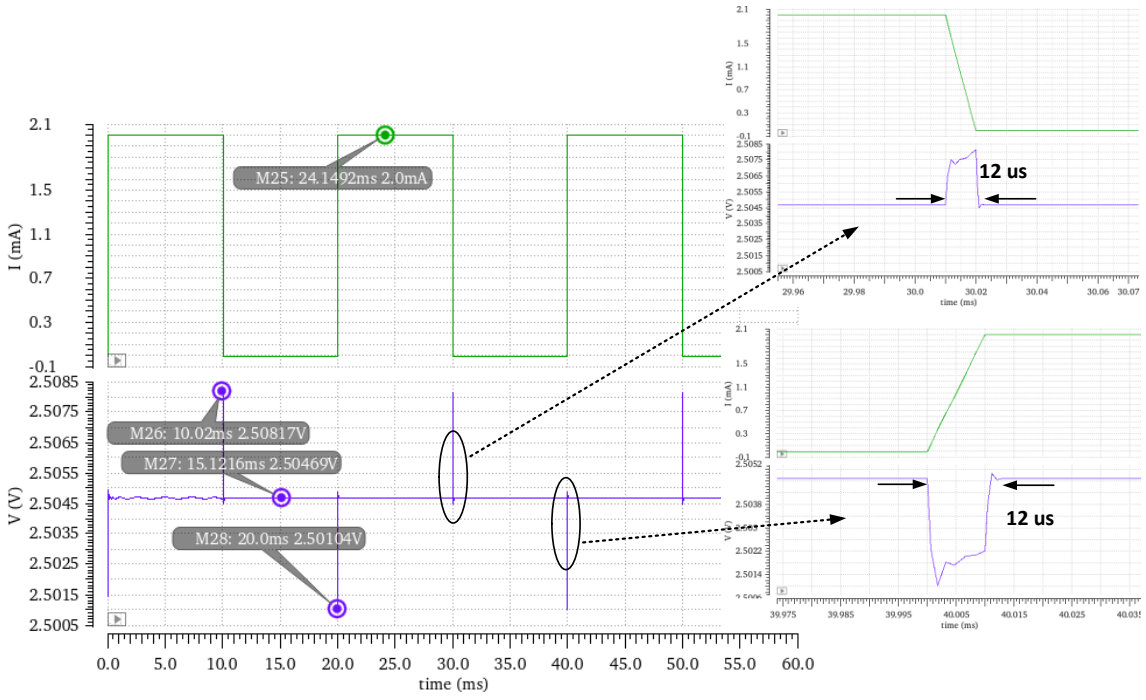


Figure 7.11 Output transient of the LDO with 2.5V output and 2mA load current.

7.3 HfO_2 Transistor Lifetime Extrapolation Result

This simulation provides HfO_2 gate stack lifetime which is the third constructing block of the comprehensive model discussed in Chapter 6. The simulation uses the 3D kMC algorithm combined with defect generation given by equation (6.19) [110] [111] and restated here,

$$r_{defect} = \nu \cdot e^{-\frac{E_a - p_{eff} \cdot \frac{2+\varepsilon}{3} \xi_{\text{programming}}}{K_B T} \cdot t} \tag{7.1}$$

Even though there are other defect generation rate studies [70] [72] in the literature, this rate is selected because it connects defect generation rate with external electric filed which relates to programming voltage.

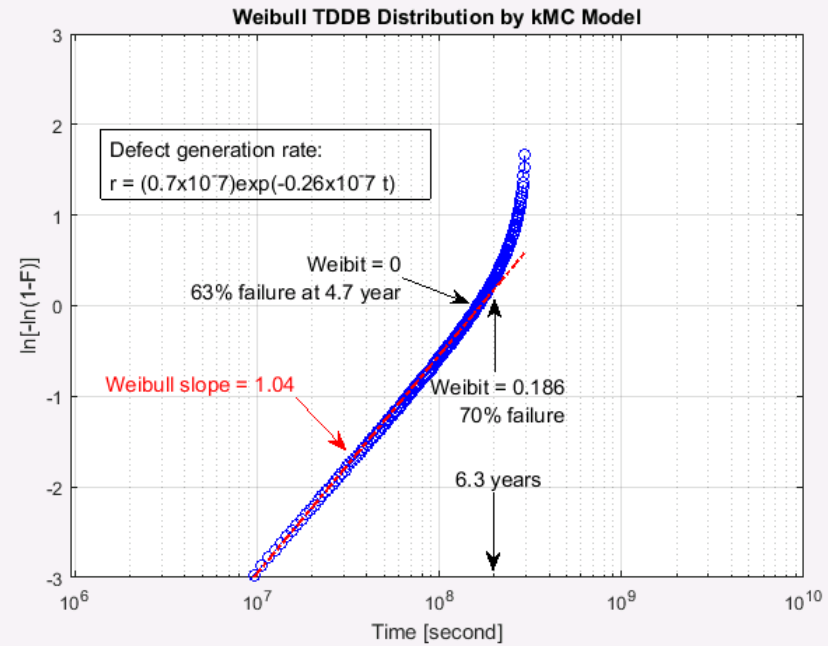


Figure 7.12 Simulated Weibull TDDB distribution of 32nm HfO_2 HK gate stack lifetime with respect to programming voltage.

As discussed in Chapter 6, section 6.3.1, HK layer breakdown first is assumed. Based on 0.6nm defect size [70] [71] and a 4nm thick HfO_2 HK layer, the dielectric bulk is divided into 6 layers in the simulation. Also, the dielectric bulk is divided into 30 segments along channel direction and

30 segments along width direction. The means the HK bulk is divided into 5400 cubes. Each cube represents a site that has the potential to become defective according to the defect generation rate specified by equation (7.1). Total 200 sample run is simulated. Each sample represents a transistor. Total are 1080000 cubs representing defect sites. Figure 7.12 shows the Weibull distribution of the gate stack. The 63% failure is extrapolated to be about 4.7 years. The 70% failure is extrapolated to be about 6.3 years.

The programming voltage can be calculated by equating equation (7.1) to the simulation line fit equation

$$rate = 0.9 \times 10^{-7} e^{-0.26\times10^{-7} t} \tag{7.2}$$

as

$$\frac{E_a - p_{eff} \cdot \frac{2+\varepsilon}{3} \cdot \xi_{\text{programming}}}{K_B T} = 0.26 \times 10^{-7} \tag{7.3}$$

The activation energy E_a for HfO_2 is 4.4eV [111] which is equivalent to $7.04 \times 10^{-19}\, J$, the effective diploe moment p_{eff} is 10.2eÅ [111] which is equivalent to $1.63 \times 10^{-28}\, C \cdot m$. Boltzmann constant K_B is $1.38 \times 10^{-23}\, J/K$. HfO_2 dielectric constant ε is 25. With these parameters in hand and assuming room temperature, the programming (or externally applied) electric field $\xi_{\text{programming}}$ is calculated to be 4.79MV/cm. The programming voltage is applied across 5nm "effective" gate oxide thickness which includes 4nm HfO_2 layer and ~1nm IL layer. Therefore, the programming voltage is calculated to be

$$\begin{aligned} V_{\text{programming}} &= \text{thickness} \cdot \xi_{\text{programming}} \\ &= 5\text{nm} \cdot 4.79\ \text{MV/cm} = 2.395\text{V} \end{aligned}$$

This result consists with the programming voltages predicted by [22] [23] [56]. The simulated Weibull distribution shows a curve up around Weibit of 0.45 which is not addressed by any literature publications. The upwards curving means higher defect generation rate towards the end

of device lifetime and the gate stack failling faster. We believe the reason of the upward curving is due to the simulation algorithm. Observe that when a lattice site becomes defective, there is an increase in the defect generation rates of its surrounding cells. As the defection generation rate is a function of time, the rate of increase is faster as more defects occure and as time passes. This is more obvious toward higher Weibit because many higher defect generation rate clusters are being connected. An alternate way of thinking is the gate oxide is becoming saturated with defects. The up curving of the plot is also graphically consistent with the observed Joule Heating phenomenon during thermal runaway before HBD. However, the actual physics reflected by the plot should be further investigated.

This model is also capable of providing lifetime extrapolation for 24nm and 15nm process nodes by reducing the number of cubes in the lattice. This work uses constant electric field, i.e. constant programming voltages, with scaled transistor geometry to demoenstrate the lifetime reduction due to the scaling. The simulated Weibull plots in figure 7.13 and figure 7.14 show the lifetime estimation for 24nm and 15nm transistors under the same programming condition as in figure 7.12. As the programming voltage remains the same, smaller geometry devices reveal shorter lifetime as expected. The reason is that the electric field across the dielectric is stronger for smaller devices under the same voltage. The 24nm node device lifetime simulation uses a 4 layer 3D grid to represent the HK dielectric. In addition the smaller oxdie volumes reach defect saturation levels more quickly due to smaller volumers and faster defect generation rates. Each layer consists of 20 cubes in the channel direction and 20 cubes in the transistor width direction. The number of samples is 200. The 63% failure time is simulated to be about 1.58 year as shown in figure 7.13. The 15nm node device lifetime simulation uses a 2 layer grid with 10 cubes in the channel direction and width direction respectively. The number of samples is 200 as well. The 63% failure time is simulated to be about 0.47 year as shown in figure 7.14. On the potentially

positive side as the gate volume becomes small quantum effects may come into play resulting in the gate oxide initial material being defect or near defect free.

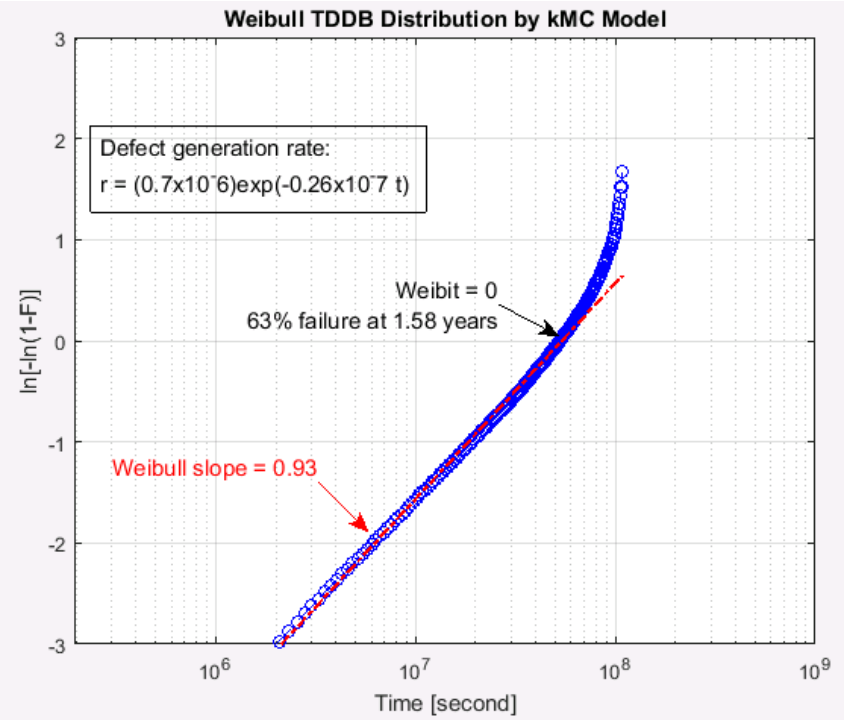


Figure 7.13 Simulated Weibull TDDB distribution of 24nm node HfO_2 HK gate stack lifetime with respect to programming voltage same as in figure 7.12.

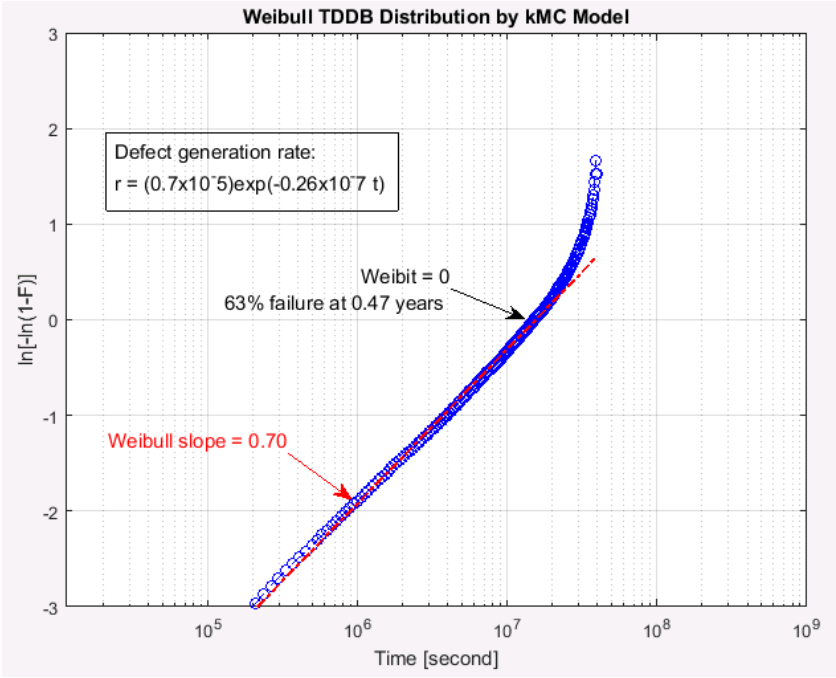


Figure 7.14 Simulated Weibull TDDB distribution of 15nm node HfO_2 HK gate stack lifetime with respect to programming voltage same as in figure 7.12.

CHAPTER VIII

CONCLUSION

The story begins from the vulnerability issues of the EEPROM used in trusted computing platforms that containing cryptographic information such as secure keys and user data. Evidence have shown the possibility of retrieving protect content by means of hardware attacks. Some of the attack methods do not require expensive tools and relatively easy to be conducted by skilled malicious users. Thus, a new EEPROM solution to the security issues for the trusted platform is in demand. At the same time, CMOS technology scaling with SiO_2 gate oxide has reached its limit for the unacceptable gate leakage, power consumption, and reliability. Therefore, HfO_2 gate dielectric material is introduced to CMOS technology. The HfO_2 certainly solves the down scaling issue but it has the special property of trapping carriers into the defects. Coincidentally, this trapping mechanism is possibly a good way to hide secure information in hardware circuits, therefore, possibly solve the trusted platform memory vulnerability issue. With this envision, we proposed the embedded EEPROM with charge trap transistors. In order to make this idea reality, two issues needs to be addressed. One is the communication interface between CPU and the charge trap based EEPROM. The other is the lifetime estimation of the EEPROM under the stress of programming voltages. This work proposed the charge trap based EEPROM core and designed an interface that achieved EEPROM core operational control and CPU communication. While the lifetime of HfO_2 dielectric transistor modeling is still an active area of research, this work builds a comprehensive model that provides data retention time and lifetime tradeoff using 3D kinetic Monte Carlo algorithm from lifetime extrapolation. Even though the model employs the defect

generation rate of HfO_2 from literature, the first step has been made for connecting lifetime and programming protocol of the charge trap based EEPROM. The future work could be refining the model when experimental lifetime data of the EEPROM is obtained and/or more widely accepted HfO_2 lifetime mode is derived.

REFERENCES

[1] A. Tomlinson, "Introduction to the TPM," *Smart Cards, Token, Security and Applications.* Springer US, 2008, pp. 155-172.
[2] http://www. trustedcomputinggroup.org
[3] S. Bajikar, "Trusted platform module (TPM) based security on notebook PCs - white paper," Mobile Platforms Group, Intel Corporation, June 20, 2002.
[4] https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-1-Architecture-01.38.pdf
[5] TCG, "TCG Specification Architecture Overview," TCG Specification Revision 1.4, The Trusted Computing Group, Portland, OR, USA, Aug. 2007.
[6] S. Skorobogatov, "How Microprobing Can Attack Encrypted Memory," *2017 Euromocro Conference on Digital System Design (DSD)*, Vienna, 2017, pp. 244-251.
[7] O. Kommerling and M. G. Kuhn, "Design principles for tamper-resistant smartcard processors," USENIX Workshop on Smartcard Technology, Chicago, Illinois, USA, May 1999.
[8] F. Beck, *Integrated Circuits Failure Analysis: A Guide to Preparation Techniques*, John Wiley & Sons, 1997.
[9] K. Kursawe, D. Schellekens, and B. Preneel, "Analyzing trusted platform communication," in Proc. ECRYPT Workshop Cryptographic Advances Secure Hardware, 2005, pp. 1-8.
[10] J. Halderman *et al*, "Lest we remember: code-boot attacks on encryption keys," *Communications of the ACM*, vol. 52(5), pp. 91-98, May 2009.
[11] O. Lo, W. J. Buchanan, and D. Carson, "Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA)," *J. Cyber Security Techn.*, vol. 1, no. 2, pp. 88-107, 2017.
[12] D. Agrawal, B. Archambeault, J. R. Rao, P. Rohatgi (2003) The EM Side-Channel(s). In: B. S. Kaliski, K. Koc, C. Paar (eds) Cryptographic Hardware and Embedded Systems - CHES 2002. CHES 2002. Lecture Notes in Computer Science, vol. 2523. Springer, Berlin, Heidelberg.
[13] JJ. Quisquater, D. Samyde (2001) ElectroMagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards. In: I. Attali, T. Jensen (eds) Smart Card Programming and Security. E-smart 2001. Lecture Notes in Computer Science, vol. 2140. Springer, Berlin, Heidelberg.
[14] D. Brumley and D. Boneh, "Remote timing attacks are practical," *the 12th Usenix Security Symposium*, 2003.
[15] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proc. CRYPTO'96*, vol. LNCS 1109, pp. 104-113, 1996.

[16] S. Skorobogatov, "Local heating attacks on Flash memory devices," *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, Francisco, CA, 2009, pp. 1-6.
[17] P. Kocher, J. Jaffe and B. Jun, "Differential power analysis," *Annual International Cryptology Conference, CRYPTO 1999: Advances in Cryptology*, pp. 388-397.
[18] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, ''Introduction to differential power analysis,'' J. Cryptogr. Eng., vol. 1, no. 1, pp. 5–27, 2011.
[19] U. Banerjee, L. Ho, and S. Koppula, "Power-based side-channel attack for aes key extraction on the atmega328 microcontroller," 2015.
[20] P. A. H. Peterson, "Cryptkeeper: Improving security with encrypted RAM," *2010 IEEE International Conference on Technologies for Homeland Security (HST)*, Waltham, MA, 2010, pp. 120-126.
[21] P. Choi and D. K. Kim, "Design of security enhanced TPM chip against invasive physical attacks," *2012 IEEE International Symposium on Circuits and Systems*, Seoul, 2012, pp. 1787-1790.
[22] F. Khan, E. Cartier, J. C. S. Woo and S. S. Iyer, "Charge Trap Transistor (CTT): An Embedded Fully Logic-Compatible Multiple-Time Programmable Non-Volatile Memory Element for High- k -Metal-Gate CMOS Technologies," in *IEEE Electron Device Letters*, vol. 38, no. 1, pp. 44-47, Jan. 2017.
[23] C. Kothandaraman et al, "Oxygen vacancy traps in Hi-K/Metal gate technologies and their potential for embedded memory applications,"2015 IEEE International Reliability Physics Symposium, Monterey, CA, pp. MY.2.1-MY.2.4., 2015.
[24] S. Mohsenifar and M. H. Shahrokhabadi, "Gate Stack High-κ Materials for Si-Based MOSFETs Past, Present, and Futures," *Microelectronics and Solid State Electronics*, vol. 4 No. 1, pp. 12-24, 2015.
[25] R. D. Clark, "Emerging Applications for High k Materials in VLSI Technology," *Materials*, vol. 7, no. 4, pp. 2913-2944, 2014.
[26] J. Robertson and R. Wallace, "High-K materials and metal gates for CMOS applications," *Materials Science and Engineering: R: Reports*, vol. 88, pp. 1-41, Feb. 2015.
[27] A. S. Oates, "Reliability issues for high-k gate dielectrics," *IEEE International Electron Devices Meeting 2003*, Washington, DC, USA, 2003, pp. 38.2.1-38.2.4.
[28] E. P. Gusev and C. P. D'Emic, "Charge detrapping in HfO_2 high-k gate dielectric stacks," Applied Physics Letters, vol. 83, no. 25, Dec. 2003, pp. 5223-5225.
[29] S. Zafar, A. Callegari, E. Gusev, and M. V. Fischetti, "Charge trapping related threshold voltage instabilities in high permittivity gate dielectric stacks," *J. App. Phys.*, vol. 93, no. 11, pp. 9298-9303, 2003.
[30] J. Robertson, "Band offsets of wide-band-gap oxides and implications for future electronic devices," *J. Vac. Sci. Techno. B*, vol. 18, iss. 3, pp. 1785-1791, 2000.
[31] G. He and Z. Sun, *High-k Gate Dielectrics for CMOS Technology*, Wiley-VCH Verlag GmbH & Co. KGaA, 2012.
[32] T. Nigam and A. Kerber, "Reliability modeling of HK MG technologies," *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*, San Jose, CA, 2014, pp. 1-8.

[33] A. Kerber and E. A. Cartier, "Reliability Challenges for CMOS Technology Qualifications With Hafnium Oxide/Titanium Nitride Gate Stacks," in *IEEE Transactions on Device and Materials Reliability*, vol. 9, no. 2, pp. 147-162, June 2009
[34] B. H. Lee, "Unified TDDB model for stacked high-k dielectrics," *2009 IEEE International Conference on IC Design and Technology*, Austin, TX, 2009, pp. 83-87.
[35] M. M. Frank, "High-k/metal gate innovations enabling continued CMOS scaling," *2011 Proceedings of the ESSCIRC (ESSCIRC)*, Helsinki, 2011, pp. 50-58.
[36] Y. Nara, IEEE/AMAT Seminar, Nov. 14 2008.
[37] A. S. Foster, F. Lopez Gejo, A. L. Shluger, and R. M. Nieminen, "Vacancy and interstitial defects in hafnia," *Phys. Rev.* B 65, pp. 174117, 2002.
[38] K. Xiong, J. Robertson, M. C. Gibson, and S. J. Clark, "Defect energy levels in HfO_2 high-dielectric-constant gate oxide," *App. Phys. Lett.*, vol. 87, iss. 18, pp. 183505, 2005.
[39] J. L. Gavartin, A. L. Shluger, A. S. Foster, and G. Bersuker, "The role of nitrogen-related defects in high-k dielectric oxides: Density-functional studies," *J. App. Phys.*, vol. 97, iss. 5, pp. 053704, 2005.
[40] K. Xiong and J. Robertson, "Point defects in HfO_2 high k gate oxide," *Microel. Engin.*, vol. 80, pp. 408-411, 2005.
[41] G. C. Jegert, *Modeling of leakage currents in high-k dielectrics*, Technische Universitat Munchen, Dissertation.
[42] W. Lee and C. Hu, "Modeling CMOS tunneling currents through ultrathin gate oxide due to conduction- and valence-band electron and hole tunneling," in *IEEE Transactions on Electron Devices*, vol. 48, no. 7, pp. 1366-1373, July 2001.
[43] S. N. Mohammad, G. Fiorenza, A. Acovic, J. B. Johnson and R. L. Carter, "Fowler-nordheim tunneling of carriers in mos transistors: two-dimensional simulation of gate current employing FIELDAY," *Solid-state Electronics*, Vol. 38, No. 4. pp. 807-814, 1995.
[44] S. Sahhaf, R. Degraeve, P. J. Roussel, B. Kaczer, T. Kauerauf and G. Groeseneken, "A New TDDB Reliability Prediction Methodology Accounting for Multiple SBD and Wear Out," in *IEEE Transactions on Electron Devices*, vol. 56, no. 7, pp. 1424-1432, July 2009.
[45] N. Raghavan, K. L. Pey, and K. Shubhakar, "High-k dielectric breakdown in nanoscale logic devices - Scientific insight and technology impact," *Microelectronics Reliability*, vol. 54, iss. 5, pp. 847-860, May 2014.
[46] F. F. Aghdam and H. Liao, "Reliability study on high-k bi-layer dielectrics," *2017 Annual Reliability and Maintainability Symposium (RAMS)*, Orlando, FL, 2017, pp. 1-7.
[47] F. Crupi, R. Degraeve, A. Kerber, D. H. Kwak and G. Groeseneken, "Correlation between Stress-Induced Leakage Current (SILC) and the HfO_2 bulk trap density in a SiO_2/HfO_2 stack," *2004 IEEE International Reliability Physics Symposium. Proceedings*, Phoenix, AZ, USA, 2004, pp. 181-187.
[48] R. O'Connor et al, "SILC defect generation spectroscopy in HfSiON using constant voltage stress and substrate hot electron injection," *International Reliability Physics Symposium*, pp. 324-329, 2008.
[49] M. Rieth and W. Schommers, *Handbook of Theoretical and Computational Nanotechnology*, vol. 10, pp. 469-543, ISBN: 1-58883-052-7.

[50] S. J. Rhee and J. C. Lee, “Threshold voltage instability characteristics of HfO_2 dielectrics n-MOSFETs,” *Microelectronics Reliability*, vol. 45, iss. 7-8, pp. 1051-1060, 2005.
[51] A. Kerber *et al*., "Origin of the threshold voltage instability in SiO_2/HfO_2 dual layer gate dielectrics," in *IEEE Electron Device Letters*, vol. 24, no. 2, pp. 87-89, Feb. 2003.
[52] Xuguang Wang and Dim-Lee Kwong, "A novel high-k SONOS memory using $TaN/Al_2O_3/Ta_2O_5/HfO_2/Si$ structure for fast speed and long retention operation," in *IEEE Transactions on Electron Devices*, vol. 53, no. 1, pp. 78-82, Jan. 2006.
[53] P. H. Tsai et al, “Novel SONOS - type nonvolatile memory device with stacked tunneling and charge trapping layers,” *Solid-state Electronics*, vol. 52, iss. 10, pp. 1573-1577, 2008.
[54] S. Maikap, “Charge trapping characteristics of atomic-layer-deposited HfO_2 films with Al_2O_3 as a blocking oxide for high-density non-volatile memory device applications,” *Semicond. Sci. Technol.*, vol. 22, no. 8, pp. 884-889, 2007.
[55] F. Khan, E. Cartier, C. Kothandaraman, J. C. Scott, J. C. S. Woo and S. S. Iyer, "The Impact of Self-Heating on Charge Trapping in High-k-Metal-Gate nFETs," in *IEEE Electron Device Letters*, vol. 37, no. 1, pp. 88-91, Jan. 2016.
[56] B. Jayaraman *et al*., "80-kb Logic Embedded High-K Charge Trap Transistor-Based Multi-Time-Programmable Memory With No Added Process Complexity," in *IEEE Journal of Solid-State Circuits*, vol. 53, no. 3, pp. 949-960, March 2018.
[57] G. Ribes *et al*., "Review on high-k dielectrics reliability issues," in *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 1, pp. 5-19, March 2005.
[58] F. Arnaud *et al*., "32nm general purpose bulk CMOS technology for high performance applications at low voltage," *2008 IEEE International Electron Devices Meeting*, San Francisco, CA, 2008, pp. 1-4.
[59] E. Cartier *et al*., "Fundamental aspects of HfO2-based high-k metal gate stack reliability and implications on t_{inv}-scaling," *2011 International Electron Devices Meeting*, Washington, DC, 2011, pp. 18.4.1-18.4.4.
[60] S. Krishnan et al., “Mechanistic understanding of breakdown and bias temperature instability in high-k metal devices using inline fast ramped bias test,” *IEEE International Reliability Physics Symposium Technical Digest*, pp. 4A.5.1-4A.5.5, 2011.
[61] G. Bersuker et al., “Mechanism of high k dielectric-induced breakdown of the interfacial SiO_2 layer,” *Proc. IEEE Int. Rel. Phys. Symp.*, pp. 373-378, 2010.
[62] E. Cartier and A. Kerber, "Stress-induced leakage current and defect generation in nFETs with HfO_2/TiN gate stacks during positive-bias temperature stress," *2009 IEEE International Reliability Physics Symposium*, Montreal, QC, 2009, pp. 486-492.
[63] C. D. Young et al., “New insights into SILC-based life time extraction,” *Proc. IEEE Int. Rel. Phys. Symp.*, pp. 5D.3.1-5D.3.5, 2012.
[64] N. Raghavan, K. L. Pey, K. Shubhakar, X. Wu, W. H. Liu and M. Bosman, "Role of grain boundary percolative defects and localized trap generation on the reliability statistics of high-κ gate dielectric stacks," *2012 IEEE International Reliability Physics Symposium (IRPS)*, Anaheim, CA, 2012, pp. 6A.1.1-6A.1.11.
[65] R. Pagano *et al*., “A novel approach to characterization of progressive breakdown in high-k/metal gate stacks,” *Microelectronics Reliability*, vol. 48, iss. 11-12, p. 1759, 2008.

[66] G. Bersuker, N. Chowdhury, C. Young, D. Heh, D. Misra and R. Choi, "Progressive Breakdown Characteristics of High-K/Metal Gate Stacks," *2007 IEEE International Reliability Physics Symposium Proceedings. 45th Annual*, Phoenix, AZ, 2007, pp. 49-54.
[67] J. W. McPherson, *Reliability Physics and Engineering Time-to-Failure Modeling*, 2nd Ed., Springer, 2013.
[68] Y. H. Kim et al., "Hard and soft-breakdown characteristics of ultrathin HfO_2 under dynamic and constant voltage stress," *IEDM Tech. Dig.*, pp. 629-632, 2002.
[69] M. Koyama et al., "Degradation mechanism of HfSiON gate insulator and effect of nitrogen composition on the statistical distribution of the breakdown," *IEDM Tech. Dig.*, pp. 849-852, 2003.
[70] T. Nigam, A. Kerber and P. Peumans, "Accurate model for time-dependent dielectric breakdown of high-k metal gate stacks," *2009 IEEE International Reliability Physics Symposium*, Montreal, QC, 2009, pp. 523-530.
[71] N. Raghavan, K. Shubhakar and K. L. Pey, "Monte Carlo evidence for need of improved percolation model for non-weibullian degradation in high-κ dielectrics," *2013 Proceedings Annual Reliability and Maintainability Symposium (RAMS)*, Orlando, FL, 2013, pp. 1-7.
[72] H. Xu et al, "Temperature- and voltage-dependent trap generation model in high-k metal gate MOS device with percolation simulation," *Chin. Phys. B*, vol. 25, no. 8, pp. 087306, 2016.
[73] S. Oh and Y. T. Yeow, "A modification to the Fowler-Nordheim tunneling current calculation for thin MOS structures", *Solid-State Electronics*, Vol. 31, Issue. 6, June 1988, pp. 1113-1118.
[74] R. J. Baker, *CMOS Circuit Design, Layout, and Simulation*, 3rd ed. IEEE Press, Wiley, 2010.
[75] Kenji Okada, Hiroyuki Ota, Toshihide Nabatame and Akira Toriumi, "TDDB and BTI reliabilities of high-k stacked gate dielectrics - Impact of initial traps in high-k layer -," *2008 IEEE International Conference on Integrated Circuit Design and Technology and Tutorial*, Austin, TX, 2008, pp. 87-90.
[76] G. D. Wilk *et al*., "Improved film growth and flatband voltage control of ALD HfO_2 and Hf-Al-O with n+ poly-Si gates using chemical oxides and optimized post-annealing," *2002 Symposium on VLSI Technology. Digest of Technical Papers (Cat. No.01CH37303)*, Honolulu, HI, USA, 2002, pp. 88-89.
[77] J. W. McPherson and H. C. Mogul, "Underlying physics of the thermochemical E model in describing low-field time-dependent dielectric breakdown in SiO_2 thin films," *J. Appl. Phys.*, vol. 84, No. 3, p. 1513, 1998.
[78] Ih-Chin Chen, S. E. Holland and Chenming Hu, "Electrical Breakdown in Thin Gate and Tunneling Oxides," in *IEEE Journal of Solid-State Circuits*, vol. 20, no. 1, pp. 333-342, Feb. 1985.
[79] K. F. Schuegraf and C. Hu, "Hole injection SiO_2 breakdown model for very low voltage lifetime extrapolation," in *IEEE Transactions on Electron Devices*, vol. 41, no. 5, pp. 761-767, May 1994.
[80] K. F. Schuegraf and C. Hu, "Hole injection oxide breakdown model for very low voltage lifetime extrapolation," *31st Annual Proceedings Reliability Physics 1993*, Atlanta, GA, USA, 1993, pp. 7-12.

[81] S. K. Lai, “Interface trap generation in silicon dioxide when electrons are captured by trapped holes,” *J. Applied Phys*., vol. 54, no. 5, pp. 2540, 1983.
[82] H. Uchida and T. Ajioka, “Electron trap center generation due to hole trapping in SiO_2 under Fowler-Nordheim tunneling stress,” *Appl. Phys. Lett.*, vol. 51, no. 6, pp.433, 1987.
[83] S. Ogaea, N. Shiono, and M. Shimaya, “Neutral electron trap generation in SiO_2 by hot holes,” *Appl. Phys. Lett.*, vol. 56, no. 14, pp. 1329, 1990.
[84] J. W. McPherson, “Time dependent dielectric breakdown physics – models revisited,” *Microelectronics Reliability*, vol. 52, pp. 1753-1760, 2012.
[85] J. R. Lloyd, "On the physical interpretation of the impact damage model in TDDB of low-k dielectrics," *2010 IEEE International Reliability Physics Symposium*, Anaheim, CA, 2010, pp. 943-946.
[86] N. Suzumura *et al*., "A New TDDB Degradation Model Based on Cu Ion Drift in Cu Interconnect Dielectrics," *2006 IEEE International Reliability Physics Symposium Proceedings*, San Jose, CA, 2006, pp. 484-489.
[87] A. Kerber, M. Rohner, T. Pompl, R. Duschl and M. Kerber, "Lifetime Prediction for CMOS Devices with Ultra Thin Gate Oxides Based on Progressive Breakdown," *2007 IEEE International Reliability Physics Symposium Proceedings. 45th Annual*, Phoenix, AZ, 2007, pp. 217-220.
[88] R. Degraeve *et al*., "Degradation and breakdown of 0.9 nm EOT SiO_2 ALD HfO_2 metal gate stacks under positive constant voltage stress," *IEEE InternationalElectron Devices Meeting, 2005. IEDM Technical Digest.*, Washington, DC, 2005, pp. 408-411.
[89] K. Okada, H. Ota, T. Nabatame and A. Toriumi, "Dielectric Breakdown in High-K Gate Dielectrics - Mechanism and Lifetime Assessment," *2007 IEEE International Reliability Physics Symposium Proceedings. 45th Annual*, Phoenix, AZ, 2007, pp. 36-43.
[90] G. Bersuker *et al*., "Breakdown in the metal/high-k gate stack: Identifying the “weak link” in the
multilayer dielectric," *2008 IEEE International Electron Devices Meeting*, San Francisco, CA, 2008, pp. 1-4.
[91] G. Ribes *et al*., "High-K gate stack breakdown statistics modeled by correlated interfacial layer and high-k breakdown path," *2010 IEEE International Reliability Physics Symposium*, Anaheim, CA, 2010, pp. 364-368.
[92] D. Choi *et al*., "Interfacial-Layer-Driven Dielectric Degradation and Breakdown of HfSiON/SiON Gate Dielectric nMOSFETs," in *IEEE Electron Device Letters*, vol. 32, no. 10, pp. 1319-1321, Oct. 2011.
[93] M. Houssa, P. W. Mertens and M. M. Heyns, “Relation between stress-induced leakage current and time-dependent dielectric breakdown in ultra-thin gate oxides,” *Semiconductor Science and Technology*, vol. 14, No. 10, pp. 892-896, 1999.
[94] A. Ojha and N. R. Mohapatra, "Trap-assisted carrier transport through the multi-stack gate dielectrics of HKMG nMOS transistors: A compact model," *2017 47th European Solid-State Device Research Conference (ESSDERC)*, Leuven, 2017, pp. 200-203.
[95] F. Jimenez-Molinos, F. Gamiz, A. Palma, P. Cartujo, and J. A. Lopez-Villanueva, “Direct and trap-assisted elastic tunneling through ultrathin gate oxides,” *J. Applied Phys*., vol. 91, No. 8, pp. 5116-5124, 2002.

[96] K. H. Gundlach and J. G. Simmons, “Range of validity of the WKB tunnel probability, and comparison of experimental data and theory,” *Thin Solid Films*, vol. 4, iss. 1, pp. 61-79, July 1969.
[97] J. H. Stathis, “Percolation models for gate oxide breakdown,” *J. Applied Phys.*, vol. 86, no. 10, pp. 5757-5766, 1999.
[98] T. Kauerauf *et al*., "Methodologies for sub-1nm EOT TDDB evaluation," *2011 International Reliability Physics Symposium*, Monterey, CA, 2011, pp. 2A.2.1-2A.2.10.
[99] Ph. J. Roussel, R. Degraeve, S. Sahhaf and G. Groeseneken, “A consistent model for the hard breakdown distribution including digital soft breakdown: the noble art of area scaling,” *Microelectronic Engieering*, vol. 84, iss. 9-10, Sep.-Oct. 2007, pp. 1925-1928.
[100] T. Wang, H. C. Ma, C. H. Li, Y. H. Lin, C. H. Chien and T. F. Lei, "Charge Retention Loss in a HfO_2 Dot Flash Memory via Thermally Assisted Tunneling," in *IEEE Electron Device Letters*, vol. 29, no. 1, pp. 109-110, Jan. 2008.
[101] B. Goll and H. Zimmermann, *Comparators in Nanometer CMOS Technology*, Springer, 2015.
[102] M. J. M. Pelgrom, *Analog-to-digital Conversion*. 3rd ed., Springer, 2010.
[103] R. Bez, E. Camerlenghi, A. Modelli and A. Visconti, "Introduction to flash memory," in *Proceedings of the IEEE*, vol. 91, no. 4, pp. 489-502, April 2003.
[104] T. Adiono, S. Harimurti and G. Meliolla, "Design and simulation of hafnium dioxide based charge trapping flash memory device," *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*, Langkawi, 2017, pp. 1-4.
[105] H. W. You and W. J. Cho, “Charge trapping properties of the HfO_2 layer with various thicknesses for charge trap flash memory applications,” *Applied Phys. Lett.*, vol. 96, iss. 9, pp. 093506, 2010.
[106] M. C. Kim, et al., “Nonvolatile memories using deep traps formed in HfO_2 by Nb ion implantation,” *J. Applied Phys.*, vol. 109, iss. 5, pp. 053703, 2011.
[107] F. Driussi, S. Spiga, A. Lamperti, G. Congedo and A. Gambi, "Simulation Study of the Trapping Properties of HfO_2 -Based Charge-Trap Memory Cells," in *IEEE Transactions on Electron Devices*, vol. 61, no. 6, pp. 2056-2063, June 2014.
[108] A. Al-Futaisi and T. W. Patzek, “Extension of Hoshen-Kopelman algorithm to non-lattice environments,” *Physica A: Statistical Mechanics and its Applications*, vol. 321, iss. 3-4, pp. 665-678, 2003.
[109] R. Sedgewick and K. Wayne, *Algorithms*, 4th ed., Addison-Wesley, 2011.
[110] S. Mei, N. Raghavan, K. Shubhakar, M. Bosman and K. L. Pey, "Multiphysics based 3D percolation framework model for multi-stage degradation and breakdown in high-κ — Interfacial layer stacks," *2016 IEEE International Reliability Physics Symposium (IRPS)*, Pasadena, CA, 2016, pp. 7A-2-1-7A-2-6.
[111] J. McPherson, J. Kim, A. Shanware, H. Mogul and J. Rodriguez, "Proposed universal relationship between dielectric breakdown and dielectric constant," *Digest. International Electron Devices Meeting,,* San Francisco, CA, USA, 2002, pp. 633-636.
[112] N. A. Chowdhury and D. Misra, “Charge trapping at deep states in Hf-silicate based high-k gate dielectrics,” *Journal of The Electrochemical Society*, 154 (2), pp. G30-G37, 2007.
[113] K. E. Sickafus, E. A. Kotomin and B. P. Uberuaga, *Radiation Effects in Solids*, NATO Science Series, Springer, pp. 1-23, 2007.

[114] B. Razavi, *Design of Analog CMOS Integrated Circuits*, TATA McGraw-Hill, 2002.

APPENDICES

Part I CPU/EEPROM Interface VHDL Code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    15:04:26 04/30/2018
-- Design Name:
-- Module Name:    top - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use ieee.std_logic_unsigned.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity top is
        Port ( clock: in STD_LOGIC;
                Reset: in STD_LOGIC;
                CPU_data_ in  : in  STD_LOGIC_VECTOR(79 downto 0);
                CPURW       : in  STD_LOGIC;
                CPU_addr    : in  STD_LOGIC_VECTOR(1 downto 0);
                CPU_data_out : out  STD_LOGIC_VECTOR(79 downto 0);
                VDD2 : out  STD_LOGIC;
```

```
        VDD2 : out STD_LOGIC;
        VERS : out  STD_LOGIC;
        VREAD : out  STD_LOGIC;
        SE : out STD_LOGIC);
end top;

architecture Behavioral of top is

--------------------------- input registers -----------------------------

        signal decoder_2_to_4_out  : std_logic_vector(3 downto 0);
        signal addren              : std_logic_vector(1 downto 0);  -- address enabling
        signal mux_2_to_1_out      : std_logic_vector(79 downto 0);
        signal cmdreset            : std_logic;

        -- Data register side signals
        signal wdcntout            : std_logic_vector(3 downto 0);
        signal decoder_4_to_16_out  : std_logic_vector(15 downto 0);
        signal dataregen           : std_logic_vector(15 downto 0);
        signal datain              : std_logic_vector(79 downto 0);  -- data from either CPU or
eeprom
        type dataregoutput_type is array(0 to 15) of std_logic_vector(79 downto 0);
        signal dataregout          : dataregoutput_type;
        signal clkctrl             : std_logic; -- data register clock input
        signal mux_16_to_1_out      : std_logic_vector(79 downto 0);
        signal En_o                : std_logic;
        signal demux_1_to_2_out1    : std_logic_vector(79 downto 0);
        signal CPUdata             : std_logic_vector(79 downto 0);
        signal trictrl             : std_logic;

        -- Counter and clock part signal
        signal clkCPU   : std_logic;
        signal wd_cnt_en   : std_logic;
        signal wdcnten  : std_logic;
        signal clkout   : std_logic;
        signal Cout     : std_logic;
        signal CPUload  : std_logic;
        signal memload  : std_logic;
        signal load     : std_logic;
        signal data_reg_clk : std_logic;
        signal or_out : std_logic;
        signal or_out_delay : std_logic;
        signal nor_out : std_logic;
        signal nor_out_delay : std_logic;
        signal initregout : std_logic_vector(79 downto 0);
        signal cmdregout : std_logic_vector(79 downto 0);
        signal prom_data_out   :  STD_LOGIC_vector(79 downto 0);

------------------------------ Finite State Machine ----------------------------

        -- state definitions
```

```
constant idle     : std_logic_vector(4 downto 0) := "00000";
constant wr_start : std_logic_vector(4 downto 0) := "01000";
constant wr_vdd2  : std_logic_vector(4 downto 0) := "01001";
constant wr_vapp  : std_logic_vector(4 downto 0) := "01010";
constant wr_done  : std_logic_vector(4 downto 0) := "01011";
constant er_start : std_logic_vector(4 downto 0) := "10000";
constant er_vers  : std_logic_vector(4 downto 0) := "10001";
constant er_vapp  : std_logic_vector(4 downto 0) := "10010";
constant er_done  : std_logic_vector(4 downto 0) := "10011";
constant rd_start : std_logic_vector(4 downto 0) := "11000";
constant rd_vread : std_logic_vector(4 downto 0) := "11001";
constant rd_vapp  : std_logic_vector(4 downto 0) := "11010";
constant wr_data_reg : std_logic_vector(4 downto 0) := "11011";
constant r_done   : std_logic_vector(4 downto 0) := "11100";
signal curr_state : std_logic_vector(4 downto 0);
signal next_state : std_logic_vector(4 downto 0);

-- timer enables and loads
signal tWR_load_1     : std_logic;
signal tWR_load_2     : std_logic;
signal tWR_load       : std_logic;
signal tWR_vdd2_en    : std_logic;
signal tWR_vapp_en    : std_logic;
signal tWR_vapp_load  : std_logic;
signal tWR_timeout_en : std_logic;
signal tER_load       : std_logic;
signal tER_vers_en    : std_logic;
signal tER_vapp_load  : std_logic;
signal tER_vapp_en    : std_logic;
signal tER_timeout_en : std_logic;
signal tRD_load_1     : std_logic;
signal tRD_load_2     : std_logic;
signal tRD_load       : std_logic;
signal tRD_vread_en   : std_logic;
signal tRD_vapp_en    : std_logic;
signal t_timeout_load : std_logic;
signal t_timeout_en   : std_logic;
signal RD_DONE        : std_logic;

-- timer counts (or counter output)
signal wr_vdd2_time : std_logic_vector(7 downto 0);
signal wr_vapp_time : std_logic_vector(7 downto 0);
signal er_vers_time : std_logic_vector(7 downto 0);
signal er_vapp_time : std_logic_vector(7 downto 0);
signal rd_vread_time: std_logic_vector(7 downto 0);
signal rd_vapp_time : std_logic_vector(7 downto 0);
signal timeout      : std_logic_vector(7 downto 0);

--timing flags
signal tWR_stab : std_logic;
signal tWR_done : std_logic;
```

```
        signal tER_stab : std_logic;
        signal tER_done : std_logic;
        signal t_out  : std_logic;
        signal tRD_stab : std_logic;
        signal tRD_done : std_logic;

        -- voltage reg enable D sides
        -- test board interaction
        signal en_wr_2p5v  : std_logic;
        signal en_er_m1p8v : std_logic;
        signal en_rd_0p7v  : std_logic;

        -- register enables signals
        signal ADDR_EN  : std_logic;
        signal ADDR_load : std_logic;
        signal ADDR_CNT : std_logic_vector (7 downto 0);
        signal R_in : std_logic;
        signal W_in : std_logic;
        signal E_in : std_logic;
        signal RD : std_logic;
        signal WR : std_logic;
        signal ER : std_logic;
        signal SER : std_logic;
        signal tWR_timeout_load : std_logic;
        signal ADDR_incre_rd : std_logic;

---------------------------- control signals -------------------------------

  signal addr_decode : std_logic_vector(255 downto 0);
        signal enable : std_logic;
        signal blk : std_logic_vector(15 downto 0);
        signal bnk : std_logic_vector(3 downto 0);
        signal data_ecc : std_logic_vector(87 downto 0);

        signal  pside          :  STD_LOGIC_VECTOR(255 downto 0);
        signal  nside          :  STD_LOGIC_VECTOR(255 downto 0);

  signal  CS_bk0   :  STD_LOGIC_VECTOR(87 downto 0);
  signal  CSbar_bk0 :  STD_LOGIC_VECTOR(87 downto 0);
  signal  N_bk0    :  STD_LOGIC_VECTOR(87 downto 0);
  signal  Nbar_bk0  :  STD_LOGIC_VECTOR(87 downto 0);
  signal  TL_bk0   :  STD_LOGIC_VECTOR(87 downto 0);
  signal  CS_bk1   :  STD_LOGIC_VECTOR(87 downto 0);
  signal  CSbar_bk1 :  STD_LOGIC_VECTOR(87 downto 0);
  signal  N_bk1    :  STD_LOGIC_VECTOR(87 downto 0);
  signal  Nbar_bk1  :  STD_LOGIC_VECTOR(87 downto 0);
  signal  TL_bk1   :  STD_LOGIC_VECTOR(87 downto 0);
  signal  CS_bk2   :  STD_LOGIC_VECTOR(87 downto 0);
  signal  CSbar_bk2 :  STD_LOGIC_VECTOR(87 downto 0);
  signal  N_bk2    :  STD_LOGIC_VECTOR(87 downto 0);
  signal  Nbar_bk2  :  STD_LOGIC_VECTOR(87 downto 0);
```

```
  signal  TL_bk2    :  STD_LOGIC_VECTOR(87 downto 0);
  signal  CS_bk3    :  STD_LOGIC_VECTOR(87 downto 0);
  signal  CSbar_bk3 :  STD_LOGIC_VECTOR(87 downto 0);
  signal  N_bk3     :  STD_LOGIC_VECTOR(87 downto 0);
  signal  Nbar_bk3  :  STD_LOGIC_VECTOR(87 downto 0);
  signal  TL_bk3    :  STD_LOGIC_VECTOR(87 downto 0);

------------------------------- EEPROM signals --------------------------------

        signal  data_out            : STD_LOGIC_VECTOR(79 downto 0);
        signal  databar_out   : STD_LOGIC_VECTOR(79 downto 0);

        type mem_type is array(0 to 255) of std_logic_vector(87 downto 0);
        signal prom_bk0: mem_type;
        signal prom_bk1: mem_type;
        signal prom_bk2: mem_type;
        signal prom_bk3: mem_type;

        signal data_out_ecc_bk0: std_logic_vector(87 downto 0);
        signal databar_out_ecc_bk0: std_logic_vector(87 downto 0);
        signal data_out_ecc_bk1: std_logic_vector(87 downto 0);
        signal databar_out_ecc_bk1: std_logic_vector(87 downto 0);
        signal data_out_ecc_bk2: std_logic_vector(87 downto 0);
        signal databar_out_ecc_bk2: std_logic_vector(87 downto 0);
        signal data_out_ecc_bk3: std_logic_vector(87 downto 0);
        signal databar_out_ecc_bk3: std_logic_vector(87 downto 0);

        signal data_out_bk0: std_logic_vector(79 downto 0);
        signal databar_out_bk0: std_logic_vector(79 downto 0);
        signal data_out_bk1: std_logic_vector(79 downto 0);
        signal databar_out_bk1: std_logic_vector(79 downto 0);
        signal data_out_bk2: std_logic_vector(79 downto 0);
        signal databar_out_bk2: std_logic_vector(79 downto 0);
        signal data_out_bk3: std_logic_vector(79 downto 0);
        signal databar_out_bk3: std_logic_vector(79 downto 0);

        signal state_bk0 : std_logic_vector(7 downto 0);
        signal state_bk1 : std_logic_vector(7 downto 0);
        signal state_bk2 : std_logic_vector(7 downto 0);
        signal state_bk3 : std_logic_vector(7 downto 0);

        constant w_bk0 : std_logic_vector(7 downto 0) := "00000001"; --x01
        constant r_bk0 : std_logic_vector(7 downto 0) := "00000010"; --x02
        constant e_bk0 : std_logic_vector(7 downto 0) := "00000011"; --x03
        constant w_bk1 : std_logic_vector(7 downto 0) := "00000100"; --x04
        constant r_bk1 : std_logic_vector(7 downto 0) := "00001000"; --x08
        constant e_bk1 : std_logic_vector(7 downto 0) := "00001100"; --x0c
        constant w_bk2 : std_logic_vector(7 downto 0) := "00010000"; --x10
        constant r_bk2 : std_logic_vector(7 downto 0) := "00100000"; --x20
        constant e_bk2 : std_logic_vector(7 downto 0) := "00110000"; --x30
        constant w_bk3 : std_logic_vector(7 downto 0) := "01000000"; --x40
```

```
        constant r_bk3 : std_logic_vector(7 downto 0) := "10000000"; --x80
        constant e_bk3 : std_logic_vector(7 downto 0) := "11000000"; --xc0
        signal p_encode: std_logic_vector(7 downto 0);
        signal n_encode: std_logic_vector(7 downto 0);
        signal nside_mask : std_logic_vector(255 downto 0);

  -- used in EEPROM model
        function encode_8bit(
                in_vec: std_logic_vector(255 downto 0)
                )
                return std_logic_vector is
                variable enc: std_logic_vector(7 downto 0) := "00000000";

        begin
                for i in 0 to 255 loop
                        if in_vec(i) = '1' then
                                enc := enc or std_logic_vector(to_unsigned(i, 8));
                        end if;
                end loop;
                return enc;
        end function;
begin

---------------------------- input registers ------------------------------
-- CPU, Initial. Reg., CMD Reg. part of the circuit
U_decoder_2_to_4_1: entity work.decoder_2_to_4(decoder_2_to_4bhv)
        port map(
                a => CPU_addr,
                b => decoder_2_to_4_out);

U_Register_CE_1: entity work.Register_CE(Registerbhv)
        generic map(width => 8)
        port map(
                CE => addren(1),
                D => CPU_data_in(79 downto 72),   -- BUSY bit is the [71] of command register
                Q => cmdregout(79 downto 72),     -- command register
                Clock => clock,
                Reset => Reset);

U_Register_CE_4: entity work.Register_CE(Registerbhv)  -- Isolate [71] bit for BUSY since
using
        generic map(width => 71)                        -- different reset logic
        port map(
                CE => addren(1),
                D => CPU_data_in(70 downto 0),
                Q => cmdregout(70 downto 0),
                Clock => clock,
                Reset => Reset);

U_Register1bit: entity work.register1bit(register1bitBhv) -- 'Ready' bit register use different reset
logic
```

```
        port map(                               -- separate this 1st bit
                CE => addren(1),
                D => CPU_data_in(71),
                Q => cmdregout(71),
                Clock => clock,
                Reset => cmdreset);

        cmdreset <= RD_DONE or Reset;

U_Register_CE_2: entity work.Register_CE(Registerbhv)
        generic map(width => 80)
        port map(
                CE => addren(0),
                D => CPU_data_in(79 downto 0),
                Q => initregout(79 downto 0), -- initialization register
                Clock => clock,
                Reset => Reset);

U_mux_2_to_1_1: entity work.mux_2_to_1(mux_2_to_1bhv)
        generic map(width => 80)
        port map(
                sel => CPU_addr(0),
                in0 => initregout,
                in1 => cmdregout,
                output => mux_2_to_1_out);

        addren(0) <= decoder_2_to_4_out(0) and CPURW;
        addren(1) <= decoder_2_to_4_out(1) and CPURW;

-- Data register part of the citcuit
U_decoder_4_to_16: entity work.decoder_4_to_16(decoder_4_to_16bhv)
        port map(
                a => wdcntout,
                b => decoder_4_to_16_out);

        -- 16 data registers and enable signals
        gen_dataregen: for i in 0 to 15 generate
                dataregen(i) <= (decoder_2_to_4_out(2) or cmdregout(71)) and
decoder_4_to_16_out(i);

U_Register_CE_3: entity work.Register_CE(Registerbhv)
        generic map(width => 80)
        port map(
                CE => dataregen(i),
                D => datain,
                Q => dataregout(i),
                Clock => clkctrl,
                Reset => Reset);
end generate;

U_mux_16_to_1: entity work.mux_16_to_1(mux_16_to_1bhv)
```

```
        generic map(width => 80)
        port map(
                sel => wdcntout,
                in0 => dataregout(0),
                in1 => dataregout(1),
                in2 => dataregout(2),
                in3 => dataregout(3),
                in4 => dataregout(4),
                in5 => dataregout(5),
                in6 => dataregout(6),
                in7 => dataregout(7),
                in8 => dataregout(8),
                in9 => dataregout(9),
                in10 => dataregout(10),
                in11 => dataregout(11),
                in12 => dataregout(12),
                in13 => dataregout(13),
                in14 => dataregout(14),
                in15 => dataregout(15),
                output => mux_16_to_1_out);

U_demux_1_to_2: entity work.demux_1_to_2(demux_1_to_2bhv)
        generic map(width => 80)
        port map(
                En => En_o,
                sel => cmdregout(71),
                input => mux_16_to_1_out,
                out0 => demux_1_to_2_out1,
                out1 => prom_data_out);

U_tristate: entity work.tristate(tristateBhv)
        generic map(width => 80)
        port map(
                input => CPUdata,
                output => CPU_data_out,
                en => trictrl);

U_mux_2_to_1_3: entity work.mux_2_to_1(mux_2_to_1bhv)
        generic map(width => 80)
        port map(
                sel => CPU_addr(1),
                in0 => mux_2_to_1_out,
                in1 => demux_1_to_2_out1,
                output => CPUdata);

-- tristate control logic
        trictrl <= ((not CPURW) and decoder_2_to_4_out(0)) or ((not CPURW) and
decoder_2_to_4_out(1)) or ((not CPURW) and decoder_2_to_4_out(2));

-- En_o logic
```

```
        En_o <= ((not R_in) and (not E_in) and (not CPURW)) or ((not R_in) and (W_in or
E_in));

U_mux_2_to_1b_3: entity work.mux_2_to_1_1b(mux_2_to_1_1bBhv)
        port map(
                sel => cmdregout(71),
                in0 => clkCPU,
                in1 => tRD_done,
                output => data_reg_clk);

        clkctrl <= (not En_o) and data_reg_clk;

U_mux_2_to_1_2: entity work.mux_2_to_1(mux_2_to_1bhv)
        generic map(width => 80)
        port map(
                sel => cmdregout(71),
                in0 => CPU_data_in,
                in1 => data_out,  --from memory modle mux output
                output => datain);

-- clock logic
U_mux_2_to_1b_1: entity work.mux_2_to_1_1b(mux_2_to_1_1bBhv)
        port map(
                sel => cmdregout(71),
                in0 => clkCPU,
                in1 => clock,
                output => clkout);

        clkCPU <= clock and decoder_2_to_4_out(2);

-- counter enable
U_mux_2_to_1b_4: entity work.mux_2_to_1_1b(mux_2_to_1_1bBhv)
        port map(
                sel => cmdregout(71),
                in0 => decoder_2_to_4_out(2),
                in1 => wdcnten,
                output => wd_cnt_en);

        wdcnten <= tWR_done or tRD_done;

-- counter load
U_mux_2_to_1b_2: entity work.mux_2_to_1_1b(mux_2_to_1_1bBhv)
        port map(
                sel => cmdregout(71),
                in0 => CPUload,
                in1 => memload,
                output => load);

        or_out <= wdcntout(0) or wdcntout(1) or wdcntout(2) or wdcntout(3);

        nor_out <= ((wdcntout(0) nor wdcntout(1)) nor wdcntout(2)) nor wdcntout(3);
```

```
U_Register1bit_2: entity work.register1bit(register1bitBhv)
	port map(
		CE => '1',
		D => or_out,
		Q => or_out_delay,
		Clock => clkCPU,
		Reset => Reset);

	CPUload <= decoder_2_to_4_out(2) nand or_out_delay;

	memload <= nor_out and Cout;
----

-- The word counter
U_wd_counterv2: entity work.wdcounterv2(wdcounterv2Bhv)
	port map(
		countin => cmdregout(79 downto 76),
		countout => wdcntout,
		load => load,
		en => wd_cnt_en,
		reset => Reset,
		clk => clock,
		cout => Cout);

---------------------------------- FSM -----------------------------------

	U_fsm_reg : entity work.register_vs(register_vsBhv)
		generic map(width => 5)
		port map(
			D => next_state,
			Q => curr_state,
			CE => '1',
			CLK => clock,
			RESET => Reset);
	next_state <=
		-- wirte sequence
		wr_start when
		(
			(curr_state=idle and (cmdregout(79 downto 72)=x"f1" or cmdregout(79
downto 72)=x"01") and cmdregout(71)='1')
		 )else
		 wr_vdd2 when
		 (
			((curr_state=wr_start) or (curr_state=wr_vdd2 and tWR_stab='0'))

		 )else
		 wr_vapp when
		 (
			(curr_state=wr_vdd2 and tWR_stab='1') or
			(curr_state=wr_vapp and tWR_done='0') or
```

```
                (curr_state=wr_done and Cout='0')
            )else
            wr_done when
            (
                    (curr_state=wr_vapp and tWR_done='1') or
                    (Cout='1' and t_out='0')
            )else

            -- erase sequence
            er_start when
            (
                    (curr_state=idle and (cmdregout(79 downto 72)=x"f2" or
cmdregout(79 downto 72)=x"02") and cmdregout(71)='1')
            )else
            er_vers when
            (
                    ((curr_state=er_start) or(curr_state=er_vers and tER_stab='0'))
            )else
            er_vapp when
            (
                    (curr_state=er_vers and tER_stab='1') or
                    (curr_state=er_vapp and tER_done='0')
            )else
            er_done when
            (
                    (tER_done='1' and t_out='0')
            )else

            --read sequence
            rd_start when
            (
                    (curr_state=idle and (cmdregout(79 downto 72)=x"f3" or
cmdregout(79 downto 72)=x"03") and cmdregout(71)='1') or
                     (curr_state=wr_done and t_out='1' and Cout = '1') or
                     (curr_state=er_done and t_out='1')
            )else
            rd_vread when
            (
                    ((curr_state=rd_start) or (curr_state=rd_vread and tRD_stab='0'))
            )else
            rd_vapp when
            (
                    (curr_state = rd_vread and tRD_stab = '1') or
                    (curr_state = rd_vapp and tRD_done = '0') or
                    (curr_state= r_done and Cout='0' and cmdregout(71)='1')
            )else
            r_done when
            (
                    (curr_state = rd_vapp and tRD_done = '1')
            )else
            idle;
```

```
	-------------------------------------
	-- Timer loads/enables
	-- load wr timers in wr_start
	-- "01000"
	tWR_load_1 <= (not next_state(4)) and next_state(3) and (not next_state(2)) and
					(not next_state(1)) and (not next_state(0));

	-- en write vdd2 timer
	-- "01001"
	tWR_vdd2_en <= (not next_state(4)) and next_state(3) and (not next_state(2)) and
					(not next_state(1)) and next_state(0);


	-- en write vapp timer
	-- "01010"
	tWR_vapp_en <= (not next_state(4)) and next_state(3) and (not next_state(2)) and
					next_state(1) and (not next_state(0));

	-- "01011"
	tWR_timeout_en <= (not next_state(4)) and next_state(3) and (not next_state(2)) and
					next_state(1) and next_state(0) AND Cout;

	-- re-load voltage application timer for next word at "01011" write_done state
	tWR_load_2 <= (not next_state(4)) and next_state(3) and (not next_state(2)) and
					next_state(1) and next_state(0) and (not Cout);

	tWR_load <= tWR_load_1 or tWR_load_2;

	tWR_vapp_load <= tWR_load or tRD_load_1; -- OR with tRD_load_1 to bring
tWR_done back to 0

	------------------
	-- load erase timers in er_start = "10000"
	tER_load <= next_state(4) and (not next_state(3)) and (not next_state(2)) and
					(not next_state(1)) and (not next_state(0));

	-- enable er vers timer
	-- "10001"
	tER_vers_en <= next_state(4) and (not next_state(3)) and (not next_state(2)) and
					(not next_state(1)) and next_state(0);

	-- en er vapp timer
	-- "10010"
	tER_vapp_en <= next_state(4) and (not next_state(3)) and (not next_state(2)) and
					next_state(1) and (not next_state(0));

	-- "10011"
	tER_timeout_en <= next_state(4) and (not next_state(3)) and (not next_state(2)) and
					next_state(1) and next_state(0);
```

```
	tER_vapp_load <= tER_load or tRD_load_1; -- OR with tRD_load_1 to bring tER_done
back to 0
	--------------------
	-- timeout timer enable and load signals
	t_timeout_en <= tWR_timeout_en or tER_timeout_en;

	--t_timeout_load <= tWR_load_2 or tER_load_2 OR tRD_load_1;
	t_timeout_load <= tWR_load_1 or tER_load;

	-- load read timer in rd_start = "11000"
	tRD_load_1 <= next_state(4) and next_state(3) and (not next_state(2)) and
					(not next_state(1)) and (not next_state(0));

	-- en read vread timer = "11001"
	tRD_vread_en <= next_state(4) and next_state(3) and (not next_state(2)) and
					(not next_state(1)) and next_state(0);

	-- en read vapp timer = "11010"
	tRD_vapp_en <= next_state(4) and next_state(3) and (not next_state(2)) and
					next_state(1) and not(next_state(0));

	-- enable read done for RD_DONE state
 RD_DONE <= curr_state(4) and curr_state(3) and curr_state(2) and
	  not(curr_state(1)) and not(curr_state(0)) and Cout;

	-- re-load voltage application timer for next word at "11100" state
	tRD_load_2 <= next_state(4) and next_state(3) and next_state(2) and (not next_state(1))
					and (not next_state(0));

	tRD_load <= tRD_load_1 or tRD_load_2;

	-------------------

	-- W, E, R signals
	-- R is high during "11001", "11010", "11100"
	R_in <= next_state(4) and next_state(3) and (next_state(2) xor next_state(1) xor
next_state(0));

	-- E is high during "10001", "10010", "10011"
	E_in <= next_state(4) and (not next_state(3)) and (not next_state(2)) and
				(next_state(1) or next_state(0));

	-- W is high during "01001", "01010", "01011"
	W_in <= (not next_state(4)) and next_state(3) and (not next_state(2)) and
				(next_state(1) or next_state(0));

	-- Read application state "11010"
	RD <= curr_state(4) and curr_state(3) and (not curr_state(2)) and curr_state(1) and (not
curr_state(0));
	-- Write application state "01010"
```

```
	WR <= (not curr_state(4)) and curr_state(3) and (not curr_state(2)) and curr_state(1) and
(not curr_state(0));
	-- Erase application state "10010"
	ER <= curr_state(4) and (not curr_state(3)) and (not curr_state(2)) and curr_state(1) and
(not curr_state(0));

	-------------------------------------------------------
	-- Timers
	--write voltage stable time counter
	U_tWR_vdd2_count: entity work.counter(counterBhv)
		generic map(width => 8)
		port map(
			countin => cmdregout(19 downto 12),
			countout => wr_vdd2_time,
			load => tWR_load_1,
			en => tWR_vdd2_en,
			clock => clock,
			reset => Reset);

	tWR_stab <= (
		(not wr_vdd2_time(7)) and (not wr_vdd2_time(6)) and
		(not wr_vdd2_time(5)) and (not wr_vdd2_time(4)) and
		(not wr_vdd2_time(3)) and (not wr_vdd2_time(2)) and
		(not wr_vdd2_time(1)) and (not wr_vdd2_time(0))
  );

	--Write voltage application time counter
	U_tWR_vapp_counter: entity work.counter(counterBhv)
		generic map(width => 8)
		port map(
			countin => cmdregout(27 downto 20),
			countout => wr_vapp_time,
			load => tWR_vapp_load,
			en => tWR_vapp_en,
			clock => clock,
			reset => Reset);

	tWR_done <= (
		(not wr_vapp_time(7)) and (not wr_vapp_time(6)) and
		(not wr_vapp_time(5)) and (not wr_vapp_time(4)) and
		(not wr_vapp_time(3)) and (not wr_vapp_time(2)) and
		(not wr_vapp_time(1)) and (not wr_vapp_time(0))
  );

	--Erase voltage stable time counter
	U_tER_vers_count: entity work.counter(counterBhv)
		generic map(width => 8)
		port map(
			countin => cmdregout(35 downto 28),
			countout => er_vers_time,
			load => tER_load,
```

```
                    en => tER_vers_en,
                    clock => clock,
                    reset => Reset);

        tER_stab <= (
                (not er_vers_time(7)) and (not er_vers_time(6)) and
                (not er_vers_time(5)) and (not er_vers_time(4)) and
                (not er_vers_time(3)) and (not er_vers_time(2)) and
                (not er_vers_time(1)) and (not er_vers_time(0))
);

         --Erase voltage application time counter
         U_tER_vapp_count: entity work.counter(counterBhv)
                generic map(width => 8)
                port map(
                        countin => cmdregout(43 downto 36),
                        countout => er_vapp_time,
                        load => tER_vapp_load,
                        en => tER_vapp_en,
                        clock => clock,
                        reset => Reset);

        tER_done <= (
                (not er_vapp_time(7)) and (not er_vapp_time(6)) and
                (not er_vapp_time(5)) and (not er_vapp_time(4)) and
                (not er_vapp_time(3)) and (not er_vapp_time(2)) and
                (not er_vapp_time(1)) and (not er_vapp_time(0))
);

        -- Timeout timer for wirte and erase
        U_t_timeout_count: entity work.counter(counterBhv)
                generic map(width => 8)
                port map(
                        countin => cmdregout(67 downto 60),
                        countout => timeout,
                        load => t_timeout_load,
                        en => t_timeout_en,
                        clock => clock,
                        reset => Reset);

        t_out <= (
                        (not timeout(7)) and (not timeout(6)) and
                        (not timeout(5)) and (not timeout(4)) and
                        (not timeout(3)) and (not timeout(2)) and
                        (not timeout(1)) and (not timeout(0))
        );

        --Read voltage stable time counter
        U_tRD_vread_count: entity work.counter(counterBhv)
                generic map(width => 8)
                port map(
```

```
                countin => cmdregout(51 downto 44),
                countout => rd_vread_time,
                load => tRD_load_1,
                en => tRD_vread_en,
                clock => clock,
                reset => Reset);

        tRD_stab <= (
                (not rd_vread_time(7)) and (not rd_vread_time(6)) and
                (not rd_vread_time(5)) and (not rd_vread_time(4)) and
                (not rd_vread_time(3)) and (not rd_vread_time(2)) and
                (not rd_vread_time(1)) and (not rd_vread_time(0))
);

        --Read voltage application time counter
        U_tRD_vapp_count: entity work.counter(counterBhv)
                generic map(width => 8)
                port map(
                        countin => cmdregout(59 downto 52),
                        countout => rd_vapp_time,
                        load => tRD_load,
                        en => tRD_vapp_en,
                        clock => clock,
                        reset => Reset);

        tRD_done <= (
                (not rd_vapp_time(7)) and (not rd_vapp_time(6)) and
                (not rd_vapp_time(5)) and (not rd_vapp_time(4)) and
                (not rd_vapp_time(3)) and (not rd_vapp_time(2)) and
                (not rd_vapp_time(1)) and (not rd_vapp_time(0))
);

        SER <= (
                (not rd_vapp_time(7)) and (not rd_vapp_time(6)) and
                (not rd_vapp_time(5)) and (not rd_vapp_time(4)) and
                (not rd_vapp_time(3)) and (not rd_vapp_time(2)) and
                (not rd_vapp_time(1)) and  rd_vapp_time(0)
);

        -- Address counter
        ADDR_load <= tWR_load_1 or tER_load or tRD_load_1 or Cout;

        process(Cout, tWR_done, tER_done, tRD_done)
        begin
                if(Cout='0' and (tWR_done='1' or tER_done='1' or tRD_done='1')) then
                        ADDR_EN <= '1';
                else
                        ADDR_EN <= '0';
                end if;
        end process;
```

```
U_addr_upcount: entity work.upcounter(upcounterBhv)
        generic map(width => 8)
        port map(
                countin => cmdregout(7 downto 0),
                countout => ADDR_CNT,
                load => ADDR_load,
                en => ADDR_EN,
                clock => clock,
                reset => Reset);

-------------------------------------------------------
--test board voltage application indicator signal
--apply voltage when they are high

-- write vdd2, write vapp
-- "01001", "01010"
en_wr_2p5v <= (not next_state(4)) and next_state(3) and (not next_state(2))
                                                        and (next_state(1) xor next_state(0));

-- erase vers, erase vapp
-- "10001", "10010"
en_er_m1p8v <= next_state(4) and (not next_state(3)) and (not next_state(2))
                                        and (next_state(1) xor next_state(0));

-- read vread, read vapp
-- "11001", "11010"
en_rd_0p7v <= next_state(4) and next_state(3) and (not next_state(2))
                                and (next_state(1) xor next_state(0));

U_VDD2_reg : entity work.register_1bit_vs(register_1bit_vs_Bhv)
        port map (
                D   => en_wr_2p5v,
                Q   => VDD2,
                CE  => '1',
                CLK => clock,
                RESET   => Reset);

U_VERS_reg : entity work.register_1bit_vs(register_1bit_vs_Bhv)
        port map (
                D   => en_er_m1p8v,
                Q   => VERS,
                CE  => '1',
                CLK => clock,
                RESET   => Reset);

U_VREAD_reg : entity work.register_1bit_vs(register_1bit_vs_Bhv)
        port map (
                D   => en_rd_0p7v,
                Q   => VREAD,
                CE  => '1',
                CLK => clock,
```

```
                        RESET   => Reset);

        U_SE_reg : entity work.register1bit(register1bitBhv)
                port map (
                        D => tRD_vapp_en,
                        Q => SE,
                        CE => '1',
                        Clock => clock,
                        Reset => SER);

------------------------ EEPROM control signals -------------------------

U_addr_deco_8_to_256: entity work.addr_deco_8_to_256(addr_deco_8_to_256Bhv)
        generic map(width => 8)
        port map(
                input => ADDR_CNT,
                output => addr_decode
                );

U_decoder_4_to_16_en: entity work.decoder_4_to_16_en(decoder_4_to_16_enBhv)
        port map(
                a => cmdregout(7 downto 4),
                en => enable,
                b => blk
                );

        enable <= (not RD) and (not WR) and ER;

        -- signals before level shifter to generate RS
        -- level shifter logic has taken into account
        process(RD, WR, ER)
        begin
        for i in 0 to 255 loop
                pside(i) <= ((not blk(i/16)) and ER) or (addr_decode(i) and RD) or
(addr_decode(i) and WR);

                nside(i) <= blk(i/16) or ((not addr_decode(i)) and (not ER)) or ((not WR) and
(not ER) and (not RD));

        end loop;
        end process;

        -- bank select
U_decoder_2_to_4_2: entity work.decoder_2_to_4(decoder_2_to_4bhv)
        port map(
                a => cmdregout(9 downto 8),
                b => bnk
                );

        data_ecc <= x"00" & prom_data_out;
```

```
process(bnk, ER, WR, RD)
begin
        for i in 0 to 87 loop
CS_bk0(i) <= ((not bnk(0)) and (not RD) and (not (WR and data_ecc(i))) and ER) or
                        (bnk(0) and (not RD) and (WR and data_ecc(i)) and (not ER));

CSbar_bk0(i) <= ((not bnk(0)) and (not RD) and (not (WR and (not data_ecc(i)))) and ER) or
                        (bnk(0) and (not RD) and (WR and (not data_ecc(i))) and (not ER));

CS_bk1(i) <= ((not bnk(1)) and (not RD) and (not (WR and data_ecc(i))) and ER) or
                        (bnk(1) and (not RD) and (WR and data_ecc(i)) and (not ER));

CSbar_bk1(i) <= ((not bnk(1)) and (not RD) and (not (WR and (not data_ecc(i)))) and ER) or
                        (bnk(1) and (not RD) and (WR and (not data_ecc(i))) and (not ER));

CS_bk2(i) <= ((not bnk(2)) and (not RD) and (not (WR and data_ecc(i))) and ER) or
                        (bnk(2) and (not RD) and (WR and data_ecc(i)) and (not ER));

CSbar_bk2(i) <= ((not bnk(2)) and (not RD) and (not (WR and (not data_ecc(i)))) and ER) or
                        (bnk(2) and (not RD) and (WR and (not data_ecc(i))) and (not ER));

CS_bk3(i) <= ((not bnk(3)) and (not RD) and (not (WR and data_ecc(i))) and ER) or
                        (bnk(3) and (not RD) and (WR and data_ecc(i)) and (not ER));

CSbar_bk3(i) <= ((not bnk(3)) and (not RD) and (not (WR and (not data_ecc(i)))) and ER) or
                        (bnk(3) and (not RD) and (WR and (not data_ecc(i))) and (not ER));
        end loop;
end process;

process(bnk, ER, WR, RD)
begin
        for i in 0 to 87 loop
N_bk0(i) <= ((not bnk(0)) and WR) or (WR and (not data_ecc(i))) or (bnk(0) and ER) or (bnk(0)
and RD);
Nbar_bk0(i) <= ((not bnk(0)) and WR) or (WR and data_ecc(i)) or (bnk(0) and ER) or (bnk(0)
and RD);
N_bk1(i) <= ((not bnk(1)) and WR) or (WR and (not data_ecc(i))) or (bnk(1) and ER) or (bnk(1)
and RD);
Nbar_bk1(i) <= ((not bnk(1)) and WR) or (WR and data_ecc(i)) or (bnk(1) and ER) or (bnk(1)
and RD);
N_bk2(i) <= ((not bnk(2)) and WR) or (WR and (not data_ecc(i))) or (bnk(2) and ER) or (bnk(2)
and RD);
Nbar_bk2(i) <= ((not bnk(2)) and WR) or (WR and data_ecc(i)) or (bnk(2) and ER) or (bnk(2)
and RD);
N_bk3(i) <= ((not bnk(3)) and WR) or (WR and (not data_ecc(i))) or (bnk(3) and ER) or (bnk(3)
and RD);
Nbar_bk3(i) <= ((not bnk(3)) and WR) or (WR and data_ecc(i)) or (bnk(3) and ER) or (bnk(3)
and RD);
end loop;
end process;
```

```
process(WR, ER, bnk)
begin
	for i in 0 to 87 loop
		TL_bk0(i) <= ((not WR) and (not ER)) nor ((not WR) and (not bnk(0)));
		TL_bk1(i) <= ((not WR) and (not ER)) nor ((not WR) and (not bnk(1)));
		TL_bk2(i) <= ((not WR) and (not ER)) nor ((not WR) and (not bnk(2)));
		TL_bk3(i) <= ((not WR) and (not ER)) nor ((not WR) and (not bnk(3)));
	end loop;
end process;

---------------------------------- EEPROM model ---------------------------------

--------------------- Bank 1 ------------------------

	p_encode <= encode_8bit(pside);

	nside_mask <= nside and
x"00010001000100010001000100010001000100010001000100010001000100010001";
	-- the memory loaction of the 1st address of the active erase block
	n_encode <= encode_8bit(nside_mask);

	state_bk0 <=
	w_bk0 when
	(
		(CS_bk0 /= CSbar_bk0) and (N_bk0 /= Nbar_bk0) and
		(TL_bk0 = x"ffffffffffffffffffff")
	)else
	e_bk0 when
	(
		(CS_bk0 = x"00000000000000000000") and
		(CSbar_bk0 = x"00000000000000000000") and
		(N_bk0 = x"ffffffffffffffffffff") and
		(Nbar_bk0 = x"ffffffffffffffffffff") and
	  (TL_bk0 = x"ffffffffffffffffffff")
	)else
	r_bk0 when
	(
		(CS_bk0 = x"00000000000000000000") and
		(CSbar_bk0 = x"00000000000000000000") and
		(N_bk0 = x"ffffffffffffffffffff") and
		(Nbar_bk0 = x"ffffffffffffffffffff") and
	  (TL_bk0 = x"00000000000000000000")
	);
--------------------- Bank 2 ------------------------

	state_bk1<=
	w_bk1 when
	(
		(CS_bk1 /= CSbar_bk1) and (N_bk1 /= Nbar_bk1) and
		(TL_bk1 = x"ffffffffffffffffffff")
```

```
        )else
        e_bk1 when
        (
                (CS_bk1 = x"00000000000000000000") and
                (CSbar_bk1 = x"00000000000000000000") and
                (N_bk1 = x"ffffffffffffffffffff") and
                (Nbar_bk1 = x"ffffffffffffffffffff") and
          (TL_bk1 = x"ffffffffffffffffffff")
        )else
        r_bk1 when
        (
                (CS_bk1 = x"00000000000000000000") and
                (CSbar_bk1 = x"00000000000000000000") and
                (N_bk1 = x"ffffffffffffffffffff") and
                (Nbar_bk1 = x"ffffffffffffffffffff") and
          (TL_bk1 = x"00000000000000000000")
        );
------------------------- Bank 3 ---------------------------
        state_bk2<=
        w_bk2 when
        (
                (CS_bk2 /= CSbar_bk2) and (N_bk2 /= Nbar_bk2) and
                (TL_bk2 = x"ffffffffffffffffffff")
        )else
        e_bk2 when
        (
                (CS_bk2 = x"00000000000000000000") and
                (CSbar_bk2 = x"00000000000000000000") and
                (N_bk2 = x"ffffffffffffffffffff") and
                (Nbar_bk2 = x"ffffffffffffffffffff") and
          (TL_bk2 = x"ffffffffffffffffffff")
        )else
        r_bk2 when
        (
                (CS_bk2 = x"00000000000000000000") and
                (CSbar_bk2 = x"00000000000000000000") and
                (N_bk2 = x"ffffffffffffffffffff") and
                (Nbar_bk2 = x"ffffffffffffffffffff") and
          (TL_bk2 = x"00000000000000000000")
        );
----------------------------- Bank 4 -----------------------------

        state_bk3<=
        w_bk3 when
        (
                (CS_bk3 /= CSbar_bk3) and (N_bk3 /= Nbar_bk3) and
                (TL_bk3 = x"ffffffffffffffffffff")
        )else
        e_bk3 when
        (
                (CS_bk3 = x"00000000000000000000") and
```

```
                (CSbar_bk3 = x"00000000000000000000") and
                (N_bk3 = x"ffffffffffffffffffff") and
                (Nbar_bk3 = x"ffffffffffffffffffff") and
          (TL_bk3 = x"ffffffffffffffffffff")
        )else
        r_bk3 when
        (
                (CS_bk3 = x"00000000000000000000") and
                (CSbar_bk3 = x"00000000000000000000") and
                (N_bk3 = x"ffffffffffffffffffff") and
                (Nbar_bk3 = x"ffffffffffffffffffff") and
          (TL_bk3 = x"00000000000000000000")
        );


        prom_bk_one: process(data_ecc, ADDR_CNT, n_encode, clock)
        begin
                if(clock'event and clock='1') then
                        if(state_bk0 = w_bk0) then
                                prom_bk0(to_integer(unsigned(ADDR_CNT))) <= data_ecc;
                        elsif(state_bk0 = r_bk0) then
                                data_out_ecc_bk0 <=
prom_bk0(to_integer(unsigned(ADDR_CNT)));
                                databar_out_ecc_bk0 <= not
prom_bk0(to_integer(unsigned(ADDR_CNT)));
                elsif(state_bk0 = e_bk0) then
                        for i in 0 to 15 loop
                                prom_bk0((to_integer(unsigned(n_encode)))+i) <= (others=>'0');
                        end loop;
                end if;
        end if;
        end process;

        data_out_bk0 <= data_out_ecc_bk0(79 downto 0);
        databar_out_bk0 <= databar_out_ecc_bk0(79 downto 0);

        prom_bk_two: process(data_ecc, ADDR_CNT, n_encode, clock)
        begin
                if(clock'event and clock='1') then
                        if(state_bk1 = w_bk1) then
                                prom_bk1(to_integer(unsigned(ADDR_CNT))) <= data_ecc;
                        elsif(state_bk1 = r_bk1) then
                                data_out_ecc_bk1 <=
prom_bk1(to_integer(unsigned(ADDR_CNT)));
                                databar_out_ecc_bk1 <= not
prom_bk1(to_integer(unsigned(ADDR_CNT)));
                        elsif(state_bk1 = e_bk1) then
                                for i in 0 to 15 loop
                                        prom_bk1((to_integer(unsigned(n_encode)))+i) <=
(others=>'0');
                                end loop;
```

```
                end if;
        end if;

    end process;

    data_out_bk1 <= data_out_ecc_bk1(79 downto 0);
    databar_out_bk1 <= databar_out_ecc_bk1(79 downto 0);

    prom_bk_three: process(data_ecc, ADDR_CNT, n_encode, clock)
    begin
        if(clock'event and clock='1') then
            if(state_bk2 = w_bk2) then
                prom_bk2(to_integer(unsigned(ADDR_CNT))) <= data_ecc;
            elsif(state_bk2 = r_bk2) then
                data_out_ecc_bk2 <=
prom_bk2(to_integer(unsigned(ADDR_CNT)));
                databar_out_ecc_bk2 <= not
prom_bk2(to_integer(unsigned(ADDR_CNT)));
            elsif(state_bk2 = e_bk2) then
                for i in 0 to 15 loop
                    prom_bk2((to_integer(unsigned(n_encode)))+i) <=
(others=>'0');
                end loop;
            end if;
        end if;

    end process;

    data_out_bk2 <= data_out_ecc_bk2(79 downto 0);
    databar_out_bk2 <= databar_out_ecc_bk2(79 downto 0);

    prom_bk_four: process(data_ecc, ADDR_CNT, n_encode, clock)
    begin
        if(clock'event and clock='1') then
            if(state_bk3 = w_bk3) then
                prom_bk3(to_integer(unsigned(ADDR_CNT))) <= data_ecc;
            elsif(state_bk3 = r_bk3) then
                data_out_ecc_bk3 <=
prom_bk3(to_integer(unsigned(ADDR_CNT)));
                databar_out_ecc_bk3 <= not
prom_bk3(to_integer(unsigned(ADDR_CNT)));
            elsif(state_bk3 = e_bk3) then
                for i in 0 to 15 loop
                    prom_bk3((to_integer(unsigned(n_encode)))+i) <=
(others=>'0');
                end loop;
            end if;
        end if;

    end process;
```

```
        data_out_bk3 <= data_out_ecc_bk3(79 downto 0);
        databar_out_bk3 <= databar_out_ecc_bk3(79 downto 0);

U_4_to_1_mux_1: entity work.mux_4_to_1(mux_4_to_1Bhv)
                port map( sel0 => cmdregout(8),
                                        sel1 => cmdregout(9),
                                        in0 => data_out_bk0,
                                        in1 => data_out_bk1,
                                        in2 => data_out_bk2,
                                        in3 => data_out_bk3,
                                        output => data_out);

U_4_to_1_mux_2: entity work.mux_4_to_1(mux_4_to_1Bhv)
        port map( sel0 => cmdregout(8),
                                    sel1 => cmdregout(9),
                                    in0 => databar_out_bk0,
                                    in1 => databar_out_bk1,
                                    in2 => databar_out_bk2,
                                    in3 => databar_out_bk3,
                                    output => databar_out);

end Behavioral;
```

Part II MATLAB Code of 3D kMC Simulation

```
clear all
close all

%% Variable declaration
%% Define lattice size
column_dir_min = 1; % column represents transistor channel direction
column_dir_max = 30;

row_dir_min = 1; % row represents transistor width direction
row_dir_max = 30;

layer_min = 1; % layer represents vertical direction, perpendicular to the channel
layer_max = 6;

%% Defect generation and defect generation rate related variables
temp_defect = [];
defect = [];
coordinate = [];

defect_W = 0;
defect_L = 0;
defect_Z = 0;

initial_rate = 0. 000000045; % initial defect generation rate

rate_sum = 0; % ktot
```

```
rate_array = []; % holds defect generation rate of each lattice site
rate_array_inter = [];
rate_coordinate = [];
rate_par_sum = []; % generation rate partial sum array

rate_par_sum_index = 1;
rate_par_sum_index_max = 0;

time_elapse = 0;

index = 0;
flag = 0;
temp = 0;

%% Checking breakdown related variables
breakdown = 0;
breakdown_track = 0;

lattice = zeros(row_dir_max, column_dir_max, layer_max);

bottom_layer_cluster = 0;
top_layer_cluster = 0;

count = 0;
%% Plot related variables
target = 0;
cluster_max = 0;

B1 = [];
B2 = [];
B3 = [];

%% Number of Monte Carlo run
num_run = 100;
TTF = [];
num_sample_count = 0;
vertical_axis = [];
time_elapse_BD_one_time = 0;
BD_time = 0;

%%
c1 = 0. 00000007;
c2 = -0.000000026;

%% Code
for w = 1:num_run

    % Initialize defect generation rate at each lattice site
    % Create coordinate for each lattice site.
    % Create order array for tracking rate_array and rate_par_sum
    for z = 1:layer_max
```

```
    for y = 1:column_dir_max
        for x = 1:row_dir_max
            coordinate = [coordinate; [x y z]];
            rate_array = [rate_array; initial_rate];
        end
    end
end

while breakdown ~= 1

    %% Defect generation
    % calculate partial sum of each site defect generation rate
    for k = 1:layer_max
        for j = 1:column_dir_max
            for i = 1:row_dir_max
                index = index + 1;   % Indexing rate_array
                rate_sum = rate_sum + rate_array(index);
                rate_par_sum = [rate_par_sum; rate_sum];
            end
        end
    end

    r = rand(1,1);
    temp = r*rate_sum;

    [rate_par_sum_index_max, donot_care_3] = size(rate_par_sum);

    % Generate a defect and store coordinate in defect[]
    for i = 1:rate_par_sum_index_max
        if((temp < rate_par_sum(i+1))&(i ~= rate_par_sum_index_max))
            defect_W = coordinate(i,1);
            defect_L = coordinate(i,2);
            defect_Z = coordinate(i,3);
            flag = 1;
        elseif((temp >= rate_par_sum(i+1))&(i ~= rate_par_sum_index_max))
            defect_W = 999;
            defect_L = 999;
            defect_Z = 999;
            flag = 0;
        else
            disp('Iteration of rate partial sum array is complete');
            flag = 0;
        end

        if(flag == 1)
            break
        end
    end

    temp_defect = [defect_W defect_L defect_Z];
    defect = [defect; temp_defect];
```

```
        %disp(defect) % <---------------------->

        rd = rand(1,1);
        defect_time_temp = -log(rd)/rate_sum;
        time_elapse_BD_one_time = time_elapse_BD_one_time + defect_time_temp;
        %disp(time_elapse_BD_one_time) % <---------------------->

        % Update defect generation rate at each site
        % map convert a defect coordinate to rate_array index
        out = map(defect_W, defect_L, defect_Z, row_dir_max, column_dir_max, layer_max);
        %disp(out); % <---------------------->
        % Set the rate of newly generated defect location to zero for avoiding
        % repetitive defect generation at the same location
        rate_array(out) = 0;
        % Update 18 neighboring sites generation rate
        % 18 = face and edge touching
        % 26 = face, edge and corner touching
        % touching x-dirction faces
        out_x_1 = map(defect_W-1, defect_L, defect_Z, row_dir_max, column_dir_max,
layer_max);
        if(out_x_1 ~= out) % checks lattice boundary
           if(~ismember([defect_W-1 defect_L defect_Z], defect, 'rows')) % check if this location
already a defect
              x1 = c1*exp(c2*time_elapse_BD_one_time);
              rate_array(out_x_1) = x1;
           else
              rate_array(out_x_1) = 0;
           end
        else
           rate_array(out_x_1) = 0;
        end
        out_x_2 = map(defect_W + 1, defect_L, defect_Z, row_dir_max, column_dir_max,
layer_max);
        if(out_x_2 ~= out)
           if(~ismember([defect_W+1 defect_L defect_Z], defect, 'rows'))
              x2 = c1*exp(c2*time_elapse_BD_one_time);
              rate_array(out_x_2) = x2;
           else
              rate_array(out_x_2) = 0;
           end
        else
           rate_array(out_x_2) = 0;
        end

        % touching y-direction faces
        out_y_1 = map(defect_W, defect_L - 1, defect_Z, row_dir_max, column_dir_max,
layer_max);
        if(out_y_1 ~= out)
           if(~ismember([defect_W defect_L-1 defect_Z], defect, 'rows'))
              y1 = c1*exp(c2*time_elapse_BD_one_time);
              rate_array(out_y_1) = y1;
```

```
        else
            rate_array(out_y_1) = 0;
        end
    else
        rate_array(out_y_1) = 0;
    end
    out_y_2 = map(defect_W, defect_L + 1, defect_Z, row_dir_max, column_dir_max,
layer_max);
    if(out_y_2 ~= out)
        if(~ismember([defect_W defect_L+1 defect_Z], defect, 'rows'))
            y2 = c1*exp(c2*time_elapse_BD_one_time);
            rate_array(out_y_2) = y2;
        else
            rate_array(out_y_2) = 0;
        end
    else
        rate_array(out_y_2) = 0;
    end

    % touching z-direction faces
    out_z_1 = map(defect_W, defect_L, defect_Z - 1, row_dir_max, column_dir_max,
layer_max);
    if(out_z_1 ~= out)
        if(~ismember([defect_W defect_L defect_Z-1], defect, 'rows'))
            z1 = c1*exp(c2*time_elapse_BD_one_time);
            rate_array(out_z_1) = z1;
        else
            rate_array(out_z_1) = 0;
        end
    else
        rate_array(out_z_1) = 0;
    end
    out_z_2 = map(defect_W, defect_L, defect_Z + 1, row_dir_max, column_dir_max,
layer_max);
    if(out_z_2 ~= out)
        if(~ismember([defect_W defect_L defect_Z+1], defect, 'rows'))
            z2 = c1*exp(c2*time_elapse_BD_one_time);
            rate_array(out_z_2) = z2;
        else
            rate_array(out_z_2) = 0;
        end
    else
        rate_array(out_z_2) = 0;
    end

    % touching top edges
    out_top_1 = map(defect_W - 1, defect_L, defect_Z + 1, row_dir_max, column_dir_max,
layer_max);
    if(out_top_1 ~= out)
        if(~ismember([defect_W-1 defect_L defect_Z+1], defect, 'rows'))
            top1 = c1*exp(c2*time_elapse_BD_one_time);
```

```
            rate_array(out_top_1) = top1;
        else
            rate_array(out_top_1) = 0;
        end
    else
        rate_array(out_top_1) = 0;
    end
    out_top_2 = map(defect_W, defect_L + 1, defect_Z + 1, row_dir_max, column_dir_max,
layer_max);
    if(out_top_2 ~= out)
        if(~ismember([defect_W defect_L+1 defect_Z+1], defect, 'rows'))
            top2 = c1*exp(c2*time_elapse_BD_one_time);
            rate_array(out_top_2) = top2;
        else
            rate_array(out_top_2) = 0;
        end
    else
        rate_array(out_top_2) = 0;
    end
    out_top_3 = map(defect_W + 1, defect_L, defect_Z + 1, row_dir_max, column_dir_max,
layer_max);
    if(out_top_3 ~= out)
        if(~ismember([defect_W+1 defect_L defect_Z+1], defect, 'rows'))
            top3 = c1*exp(c2*time_elapse_BD_one_time);
            rate_array(out_top_3) = top3;
        else
            rate_array(out_top_3) = 0;
        end
    else
        rate_array(out_top_3) = 0;
    end
    out_top_4 = map(defect_W, defect_L - 1, defect_Z + 1, row_dir_max, column_dir_max,
layer_max);
    if(out_top_4 ~= out)
        if(~ismember([defect_W defect_L-1 defect_Z+1], defect, 'rows'))
            top4 = c1*exp(c2*time_elapse_BD_one_time);
            rate_array(out_top_4) = top4;
        else
            rate_array(out_top_4) = 0;
        end
    else
        rate_array(out_top_4) = 0;
    end

    % touching side edges
    out_side_1 = map(defect_W - 1, defect_L - 1, defect_Z, row_dir_max, column_dir_max,
layer_max);
    if(out_side_1 ~= out)
        if(~ismember([defect_W-1 defect_L-1 defect_Z], defect, 'rows'))
            side1 = c1*exp(c2*time_elapse_BD_one_time);
            rate_array(out_side_1) = side1;
```

```
            else
                rate_array(out_side_1) = 0;
            end
        else
            rate_array(out_side_1) = 0;
        end
        out_side_2 = map(defect_W - 1, defect_L + 1, defect_Z, row_dir_max, column_dir_max,
layer_max);
        if(out_side_2 ~= out)
            if(~ismember([defect_W-1 defect_L+1 defect_Z], defect, 'rows'))
                side2 = c1*exp(c2*time_elapse_BD_one_time);
                rate_array(out_side_2) = side2;
            else
                rate_array(out_side_2) = 0;
            end
        else
            rate_array(out_side_2) = 0;
        end
        out_side_3 = map(defect_W + 1, defect_L + 1, defect_Z, row_dir_max, column_dir_max,
layer_max);
        if(out_side_3 ~= out)
            if(~ismember([defect_W+1 defect_L+1 defect_Z], defect, 'rows'))
                side3 = c1*exp(c2*time_elapse_BD_one_time);
                rate_array(out_side_3) = side3;
            else
                rate_array(out_side_3) = 0;
            end
        else
            rate_array(out_side_3) = 0;
        end
        out_side_4 = map(defect_W + 1, defect_L - 1, defect_Z, row_dir_max, column_dir_max,
layer_max);
        if(out_side_4 ~= out)
            if(~ismember([defect_W+1 defect_L-1 defect_Z], defect, 'rows'))
                side4 = c1*exp(c2*time_elapse_BD_one_time);
                rate_array(out_side_4) = side4;
            else
                rate_array(out_side_4) = 0;
            end
        else
            rate_array(out_side_4) = 0;
        end

        % Touching bottom edges
        out_bottom_1 = map(defect_W - 1, defect_L, defect_Z - 1, row_dir_max, column_dir_max,
layer_max);
        if(out_bottom_1 ~= out)
            if(~ismember([defect_W-1 defect_L defect_Z-1], defect, 'rows'))
                bottom1 = c1*exp(c2*time_elapse_BD_one_time);
                rate_array(out_bottom_1) = bottom1;
            else
```

```
            rate_array(out_bottom_1) = 0;
        end
    else
        rate_array(out_bottom_1) = 0;
    end
    out_bottom_2 = map(defect_W, defect_L + 1, defect_Z - 1, row_dir_max, column_dir_max,
layer_max);
    if(out_bottom_2 ~= out)
        if(~ismember([defect_W defect_L+1 defect_Z-1], defect, 'rows'))
            bottom2 = c1*exp(c2*time_elapse_BD_one_time);
            rate_array(out_bottom_2) = bottom2;
        else
            rate_array(out_bottom_2) = 0;
        end
    else
        rate_array(out_bottom_2) = 0;
    end
    out_bottom_3 = map(defect_W + 1, defect_L, defect_Z - 1, row_dir_max, column_dir_max,
layer_max);
    if(out_bottom_3 ~= out)
        if(~ismember([defect_W+1 defect_L defect_Z-1], defect, 'rows'))
            bottom3 = c1*exp(c2*time_elapse_BD_one_time);
            rate_array(out_bottom_3) = bottom3;
        else
            rate_array(out_bottom_3) = 0;
        end
    else
        rate_array(out_bottom_3) = 0;
    end
    out_bottom_4 = map(defect_W, defect_L - 1, defect_Z - 1, row_dir_max, column_dir_max,
layer_max);
    if(out_bottom_4 ~= out)
        if(~ismember([defect_W defect_L-1 defect_Z-1], defect, 'rows'))
            bottom4 = c1*exp(c2*time_elapse_BD_one_time);
            rate_array(out_bottom_4) = bottom4;
        else
            rate_array(out_bottom_4) = 0;
        end
    else
        rate_array(out_bottom_4) = 0;
    end

    flag = 0;
    rate_sum = 0;
    rate_par_sum = [];
    index = 0;

    rate_coordinate = [rate_array coordinate];
    rate_coordinate = sortrows(rate_coordinate, 1);
    rate_array = rate_coordinate(:,1:1);
    coordinate = rate_coordinate(:,2:4);
```

```
%% Check breakdown
% Lattice position is a defect, set this position is 1,
% Lattice position is not a defect, set this position is 0.
for c = 1:layer_max
   for b = 1:column_dir_max
      for a = 1:row_dir_max
         if(ismember([a b c], defect, 'rows')==1)
            lattice(a,b,c) = 1;
         else
            lattice(a,b,c) = 0;
         end
      end
   end
end

% Cluster formation
% 18 = face and edge touching --> same cluster number
% 26 = face, edge and corner touching --> same cluster number
Lb = bwlabeln(lattice,18);

bottom_layer = []; % clear for next iteration
top_layer = []; % clear for next iteration

% store top and bottom layer information
for x = 1:row_dir_max
   for y = 1:column_dir_max
      if(Lb(x, y, layer_min) ~= 0)
         bottom_layer = [bottom_layer; Lb(x, y, layer_min)];
      else
         bottom_layer = bottom_layer;
      end
   end
end
for x = 1:row_dir_max
   for y = 1:column_dir_max
      if(Lb(x, y, layer_max) ~= 0)
         top_layer = [top_layer; Lb(x, y, layer_max)];
      else
         top_layer = top_layer;
      end
   end
end

% 'bottom_layer_cluster' holds the matrix size
[bottom_layer_cluster, donot_care_1] = size(bottom_layer);
% 'top_layer_cluster'' holds the matric size
[top_layer_cluster, donot_care_2] = size(top_layer);

% Check if top and bottom layers have the same cluster number.
% If it has, breakdown happens i.e. a connected or conductive path
% connecting top and bottom layer.
```

```
        if((bottom_layer_cluster >= top_layer_cluster)&(bottom_layer_cluster ~=
0)&(top_layer_cluster ~= 0))
            count = top_layer_cluster;
            for i = 1:count
                if(ismember(top_layer(i), bottom_layer, 'rows') == 1)
                    breakdown_track = breakdown_track + 1;
                else
                    breakdown_track = breakdown_track + 0;
                end
            end
        elseif((bottom_layer_cluster < top_layer_cluster)&(bottom_layer_cluster ~=
0)&(top_layer_cluster ~= 0))
            count = bottom_layer_cluster;
            for i = 1:count
                if(ismember(bottom_layer(i), top_layer, 'rows') == 1)
                    breakdown_track = breakdown_track + 1;
                else
                    breakdown_track = breakdown_track + 0;
                end
            end
        elseif((bottom_layer_cluster == 0)|(top_layer_cluster == 0))
            breakdown_track = breakdown_track + 0;
        end

        if breakdown_track > 0
            breakdown = 1;
        else
            breakdown = 0;
        end

    end

    BD_time = BD_time + time_elapse_BD_one_time;
    TTF = [TTF; BD_time];

    num_sample_count = num_sample_count + 1;
    percentage_failure = num_sample_count/num_run;
    if(percentage_failure ~= 1)
        percentage_failure = percentage_failure;
    else
        percentage_failure = 0.99;
    end

    Weibit = log(-log(1-percentage_failure));
    vertical_axis = [vertical_axis; Weibit];

    breakdown = 0;
end

%% Plot Weibull
figure(1)
```

```
semilogx(TTF, vertical_axis, 'b--o');
title('Weibull TDDB Distribution by kMC Model')
xlabel('Time [second]')
xlim([0 10^9])
ylabel('ln[-ln(1-F)]')
ylim([-3 3])
hold on
grid on

FT = polyfit(log(TTF(1:30)), vertical_axis(1:30), 1);
fitvertical_axis = polyval(FT, log(TTF));
Slope = FT(1);
Intercept = FT(2);
hold on
grid on
semilogx(TTF, fitvertical_axis, 'r-.', 'LineWidth', 1.5);

%% Plot breakdown path
figure(2)
% obtaining cluster number indicating breakdown
if((bottom_layer_cluster >= top_layer_cluster)&(bottom_layer_cluster ~= 0)&(top_layer_cluster
~= 0))
   count = top_layer_cluster;
   for i = 1:count
      if(ismember(top_layer(i), bottom_layer, 'rows') == 1)
         target = top_layer(i);
      else
         target = target;
      end
   end
elseif((bottom_layer_cluster < top_layer_cluster)&(bottom_layer_cluster ~=
0)&(top_layer_cluster ~= 0))
   count = bottom_layer_cluster;
   for i = 1:count
      if(ismember(bottom_layer(i), top_layer, 'rows') == 1)
         target = bottom_layer(i);
      else
         target = target;
      end
   end
else
   target = target;
end

% Find the max cluster number
for k = 1:layer_max
   for j = 1:column_dir_max
      for i = 1:row_dir_max
         if(Lb(i,j,k) > cluster_max)
            cluster_max = Lb(i,j,k);
         else
```

```
            cluster_max = cluster_max;
        end
    end
  end
end
% Plot defects
for t = 1:cluster_max
  for k = 1:layer_max
    for j = 1:column_dir_max
      for i = 1:row_dir_max
        if((Lb(i,j,k)~=0)&(Lb(i,j,k)==t)&(target~=t))
%           A1 = [A1;i];
%           A2 = [A2;j];
%           A3 = [A3;k];
            plot3(i,j,k,'--
o','MarkerEdgeColor','none','MarkerFaceColor',rand(1,3),'MarkerSize',11);
            xlabel('Transistor Width Direction')
            xlim([1 11])
            ylabel('Channel Direction')
            ylim([1 8])
            zlabel('Vertical Direction')
            zlim([1 7])
            grid on
            hold on
        elseif((Lb(i,j,k)~=0)&(Lb(i,j,k)==t)&(target==t))
            B1 = [B1;i];
            B2 = [B2;j];
            B3 = [B3;k];
            plot3(B1,B2,B3,'--o','Color','r','LineWidth',1.5,'MarkerSize',12);
            %plot3(B1,B2,B3,'--
o','MarkerEdgeColor','none','MarkerFaceColor','r','MarkerSize',12);
            xlabel('Transistor Width Direction')
            xlim([1 11])
            ylabel('Channel Direction')
            ylim([1 8])
            zlabel('Vertical Direction')
            zlim([1 7])
            grid on
            hold on
        end
      end
    end
  end
end

%% Prior to MATLAB R2016a version function in separate file
function output = map(x, y, z, xmax, ymax, zmax)

  if((x<1)&&(y>=1)&&(y<=ymax)&&(z>=1)&&(z<=zmax))
    output=(z-1)*ymax*xmax+(y-1)*xmax+1;
  elseif((x>xmax)&&(y>=1)&&(y<=ymax)&&(z>=1)&&(z<=zmax))
```

```
    output=(z-1)*ymax*xmax+(y-1)*xmax+xmax;
elseif((x>=1)&&(x<=xmax)&&(y<1)&&(z>=1)&&(z<=zmax))
    output=(z-1)*ymax*xmax+(1-1)*xmax+x;
elseif((x>=1)&&(x<=xmax)&&(y>ymax)&&(z>=1)&&(z<=zmax))
    output=(z-1)*ymax*xmax+(ymax-1)*xmax+x;
elseif((x>=1)&&(x<=xmax)&&(y>=1)&&(y<=ymax)&&(z<1))
    output=(1-1)*ymax*xmax+(y-1)*xmax+x;
elseif((x>=1)&&(x<=xmax)&&(y>=1)&&(y<=ymax)&&(z>zmax))
    output=(zmax-1)*ymax*xmax+(y-1)*xmax+x;
elseif((x<1)&&(y<1)&&(z>=1)&&(z<=zmax))
    output=(z-1)*ymax*xmax+(1-1)*xmax+1;
elseif((x>xmax)&&(y>ymax)&&(z>=1)&&(z<=zmax))
    output=(z-1)*ymax*xmax+(ymax-1)*xmax+xmax;
elseif((x<1)&&(y>ymax)&&(z>=1)&&(z<=zmax))
    output=(z-1)*ymax*xmax+(ymax-1)*xmax+1;
elseif((x>xmax)&&(y<1)&&(z>=1)&&(z<=zmax))
    output=(z-1)*ymax*xmax+(1-1)*xmax+xmax;
elseif((x<1)&&(y>=1)&&(y<=ymax)&&(z<1))
    output=(1-1)*ymax*xmax+(y-1)*xmax+1;
elseif((x>xmax)&&(y>=1)&&(y<=ymax)&&(z>zmax))
    output=(zmax-1)*ymax*xmax+(y-1)*xmax+xmax;
elseif((x<1)&&(y>=1)&&(y<=ymax)&&(z>zmax))
    output=(zmax-1)*ymax*xmax+(y-1)*xmax+1;
elseif((x>xmax)&&(y>=1)&&(y<=ymax)&&(z<1))
    output=(1-1)*ymax*xmax+(y-1)*xmax+xmax;
elseif((x>=1)&&(x<=xmax)&&(y<1)&&(z<1))
    output=(1-1)*ymax*xmax+(1-1)*xmax+x;
elseif((x>=1)&&(x<=xmax)&&(y>ymax)&&(z>zmax))
    output=(zmax-1)*ymax*xmax+(ymax-1)*xmax+x;
elseif((x>=1)&&(x<=xmax)&&(y<1)&&(z>zmax))
    output=(zmax-1)*ymax*xmax+(1-1)*xmax+x;
elseif((x>=1)&&(x<=xmax)&&(y>ymax)&&(z<1))
    output=(1-1)*ymax*xmax+(ymax-1)*xmax+x;
elseif((x<1)&&(y<1)&&(z<1))
    output=(1-1)*ymax*xmax+(1-1)*xmax+1;
elseif((x<1)&&(y<1)&&(z>zmax))
    output=(zmax-1)*ymax*xmax+(1-1)*xmax+1;
elseif((x<1)&&(y>ymax)&&(z<1))
    output=(1-1)*ymax*xmax+(ymax-1)*xmax+1;
elseif((x<1)&&(y>ymax)&&(z>zmax))
    output=(zmax-1)*ymax*xmax+(ymax-1)*xmax+1;
elseif((x>xmax)&&(y<1)&&(z<1))
    output=(1-1)*ymax*xmax+(1-1)*xmax+xmax;
elseif((x>xmax)&&(y<1)&&(z>zmax))
    output=(zmax-1)*ymax*xmax+(1-1)*xmax+xmax;
elseif((x>xmax)&&(y>ymax)&&(z<1))
    output=(1-1)*ymax*xmax+(ymax-1)*xmax+xmax;
elseif((x>xmax)&&(y>ymax)&&(z>zmax))
    output=(zmax-1)*ymax*xmax+(ymax-1)*xmax+xmax;
else
    output=(z-1)*ymax*xmax+(y-1)*xmax+x;
```

```
    end
end
```

VITA

Cheng Hao

Candidate for the Degree of

Doctor of Philosophy

Dissertation: A SECURE HFO2 BASED CHARGE TRAP EEPROM WITH LIFETIME AND DATA RETENTION TIME MODELING

Major Field: Electrical and Computer Engineering

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Electrical and Computer Engineering at Oklahoma State University, Stillwater, Oklahoma in July, 2019.

Completed the requirements for the Master of Science in Electrical and Computer Engineering at Oklahoma State University, Stillwater, Oklahoma in 2016.

Completed the requirements for the Bachelor of Science in Electrical and Computer Engineering at North Carolina State University, Raleigh, NC in 2011.

Experience:

Research assistant at Oklahoma State University, Jan. 2014 – May 2019
Teaching assistant at Oklahoma State University, Jan. 2014 – May 2017