CUSTOM INTERFACE FOR CHARGE TRAP EEPROM APPLICATIONS

By

VISHAL REDDY BANALA

Bachelor of Technology in Electronics and

Communication Engineering

Jawaharlal Nehru Technological University

Hyderabad, Telangana, India

2017

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2019

CUSTOM INTERFACE FOR CHARGE TRAP EEPROM APPLICATIONS

Thesis Approved:

Dr. Chriswell G. Hutchens

Thesis Advisor

Dr. Weili Zhang

Committee Chair

Dr. Ramachandra Gupta Ramakumar

Committee Member

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisor Dr. Chriswell G. Hutchens for the guidance throughout my Master’s degree. I would have definitely not reached this achievement without his support. He has been a great mentor for me since the very beginning.

I also would like to thank Dr. Ramakumar and Dr. Weili Zhang to be my committee members. I would like to thank Oklahoma State University for its excellent infrastructure and laboratories that helped me at every point.

I would like to thank my parents, Mrs. Vijaya Reddy and Mr. Ramakrishna Reddy and other family members and friends, for believing in me at all points of life. I dedicate this thesis to my parents. It is their faith and love that makes me achieve all this.

Lastly, I would like to thank my VLSI lab members, Mr. Cheng Hao and Mr. Juan Salinas, for their help throughout my thesis.

Acknowledgements reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: VISHAL REDDY BANALA

Date of Degree: MAY, 2019

Title of Study: CUSTOM INTERFACE FOR CHARGE TRAP EEPROM APPLICATIONS

Major Field: ELECTRICAL AND COMPUTER ENGINEEERING

Abstract:

Cryptographic attacks and memory observability among DRAM or PROM have drawn researcher's attention. C. Kothandaraman et al. [1] have proposed a secure memory element using charge-trap based transistor for a potential memory application. The charge trapping layer used in this work is HfO_2 which is a high-k dielectric. The properties of HfO_2 making it superior to replace traditional SiO_2 dielectric are brought together. In general, the trend of scaling non-volatile memory is presented and how dielectric thickness effects the gate length is shown using HfO_2 as a solution for scaling beyond 45nm node. This work presents an on-chip interface architecture between CPU and secure EEPROM for control and data communication. This architecture allows the secure EEPROM to be embedded with the processing unit preventing interface eavesdropping so that encryption keys can be accessed locally and securely. It is also designed to overcome the cold-boot attacks [2] and side channel attacks [3] employing on-chip implementation and parallel data communication. Based on the literature, the programming voltages are proposed to meet the optimal requirement of data retention and lifetime. Also, future predictions are made on dielectrics and device architecture for continuous scaling. The interface design is implemented in VHDL and validated with secure EEPROM model. The synthesis and simulation results of the design are presented.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

LIST OF SYMBOLS

CPU	Central Processing Unit
CMOS	Complementary Metal Oxide Semiconductor
CTM	Charge-Trapping Memory
CTL	Charge-Trapping Layer
CTT	Charge-Trap Transistor
CUI	Command User Interface
CMD[]	Command Register Bits
DRAM	Dynamic Random Access Memory
EEPROM	Electrically Erasable Programmable Read-Only Memory
EOT	Equivalent Oxide Thickness
ECC	Error Code Correction
FSM	Finite State Machine
FET	Field Effect Transistor
HfO_2	Hafnium dioxide
INIT[]	Initialization Register Bits
ITRS	International Technology Roadmap for Semiconductor
MSB	Most Significant Bit
NVM	Non-Volatile Memory
SiO_2	Silicon dioxide or Silica
SCA	Side Channel Attacks
SiN	Silicon Nitride
SA	Sense Amplifier
SNR	Signal to Noise Ratio
VHDL	VHSIC Hardware Description Language

CHAPTER I

INTRODUCTION

Encryption helps to improve data security during transferring data. DRAM and PROM are commonly used storage means for encryption keys. Post power removal, their content will not be lost instantly due to a finite discharge time [2]. The finite decay time allows retrieval of the secure content using simple techniques when memory is physically accessible. J. Halderman et al. [2] have demonstrated that DRAM loses its content over a period of seconds after cutting power or being removed from the motherboard at normal operating temperatures. O. Lo et al. [3] have successfully demonstrated side channel attacks (SCA) on serial data bus using power analysis. C. Kothandaraman et al. [1] have proposed a secure memory cell structure employing charge-trap based transistors. In addition to the secure cell structure, by embedding memory alongside the processor, it is evident that the memory is secured from physical attacks considering the memory size [4]. There is always the argument that the memory can be exploited using decrypting tools, however the memory can be secured using double layered encryption process [2] or designing a unique language tool that only a specialized processor can use to communicate with memory, in case whole chip is stolen or various threats coming along with time. In comparison with other non-volatile memory storage systems, the secure EEPROM embedded with the processing unit can be proved to be much safer and efficient.

Chapter II introduces the non-volatile memories and their applications in data security. The charge-trap transistor (CTT) and its operation is explained. Chapter II offers reasons why HfO_2 is chosen as the charge trapping layer in the CTT. The two most important attacks faced by memory are discussed. We also present how the scalability of non-volatile memory has evolved over the years. In chapter III, the architecture of interface between CPU and EEPROM or wrapper is discussed. The wrapper consists of input registers, command user interface and controller (finite state machine). Chapter IV presents the implementation of the wrapper. The write/read/erase operations are performed on the memory using VHDL as medium to commute with memory. In addition, the storage cell is explained and the four bank representation of EEPROM macro model is presented. The conclusive results and simulations are summarized in chapter V. This includes waveforms respectively write/read/erase operations as prompted by CPU and synthesis reports (timing, power and area) of the entire architecture generated via DC compiler on server. The final chapter summarizes this work explaining how safely the data can be encrypted.

CHAPTER II

EEPROM MEMORY AND ITS SECURITY ISSUES

2.1 NON-VOLATILE MEMORY (NVM)

A non-volatile memory device retains stored information in the absence of power. The dimensions of the cell are continuously down-scaled resulting in successful development of floating-gate memory over the last few decades. This has yielded high data-storage density, high program/erase speeds, low voltage operators and reduced power consumption [4]. Ever increasing fabrication density of NVM has been driven remarkably by device scaling [4]. As memory devices scale down beyond the 32 nm technology node, the technology faces significant challenges. Hence, floating-gate memory type reached its scaling limit very quickly. A relatively thick tunneling oxide and blocking layer have to be used in the floating-gate memory to maintain acceptable reliability, limiting further down-scaling of the cell size in the vertical direction [7]. In addition, maintaining a high gate coupling ratio is a major problem for the floating-gate devices down-scaling. Moreover, the number of electrons stored in floating-gate transistor decreases significantly with continuous down-scaling of the cell size. Due to these challenges, reliability factor of the floating-gate memory devices is adversely effected. As a result, some promising memory technologies have been developed for the next-generation NVM [4] [5] [7].

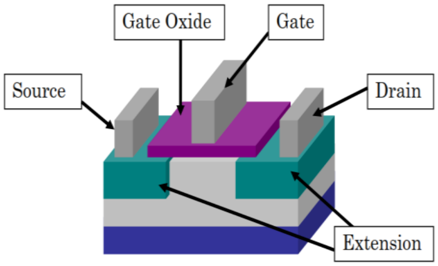


Figure 2.1 Simple structure of MOSFET [4].

2.2 ATTACKS ON MEMORY

A) Cold boot attacks

As mentioned earlier in Chapter I, DRAMs used in modern computers retain their contents for several seconds post power removal. This is possible even at room temperature and even if removed from a motherboard. When the electronics are cooled down to lower temperatures, the data will persist for minutes or even hours [2] which is more than sufficient allowing the residual data be recovered using simple and safe techniques that require only momentary physical access to the machine.

If the memory is not refreshed often or immediately erased, an attacker can extract the encryption keys with physical access to the memory. In this case, the contents persist for sufficient time to capture full-system memory images. These attacks come in three variants [2]. The first simple attack is to reboot the machine that gives the attacker access to the retained memory. The next level attack is to remove the power from the machine and then restore the power that prevents the operating system to scrub the memory before shutting down. The most effective attack is to remove the power and then transplant the memory to the attacker PC, which extracts the memory state. This attack also deprives the computer user of the option to clear the memory on boot.

Additionally, cold reboots are used to mount successful attacks on disk encryption systems, like BitLocker, TrueCrypt and FileVault, without any special assistance.

B) Side Channel Attacks

The attack on the cryptographic device carrying encryption keys is called the side channel attack. The physical output of the device is correlated with internal state of the memory to decrypt the information stored in the memory. The physical outputs can be characterized as power consumption, time taken to carry out an operation, emission of heat, light and sound [3]. Figure 2.2 shows an abstraction of this concept.

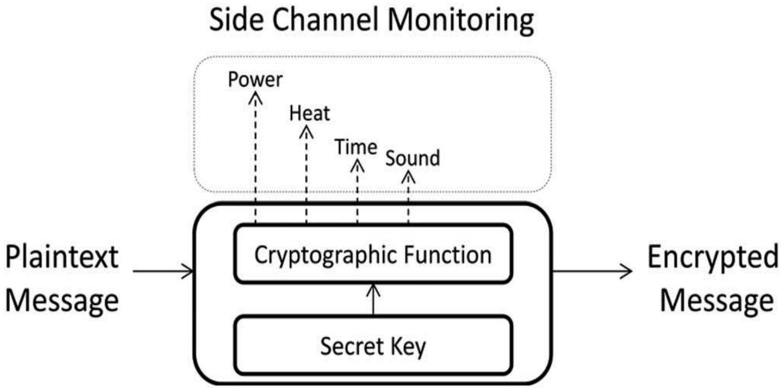


Fig 2.2 Side channel monitoring [3].

The idea is to monitor the information produced by a device during its normal operation and attempt to deduce the encryption keys, instead of trying to break the core implementation of a cryptographic device. Some of the most common attacks include timing attacks, fault attacks and power analysis [3]. With respect to our work, power analysis is more likely to decrypt the information from the secure EEPROM, assuming the chances to be insignificant. This is because of the parallel data communication, i.e., all the 80 bits of data are processed simultaneously, rather than employing one bit per cycle mechanism. In addition, the magnitude of power consumption is reduced by a significant factor by moving the memory from off-chip to on-chip alongside the processor.

2.3 CHARGE-TRAP TRANSISTOR (CTT)

Charge trapping memory (CTM) was firstly introduced in 1967 [4] [7]. CTM has some distinguished advantages over the traditional floating-gate memory. CTM defines the write/erase states via adding charges to and removing charges from the charge-storage layer respectively, similar to floating-gate memory. Apart from the floating-gate cell where charges are mainly stored in the conduction band of the floating gate, the main difference of charge-trapping flash memory is that charges are located at the interfacial layer and bulk traps distributed in the band-gap of the charge-trapping layer [5] [7].

Figure 2.3 shows the structure of a charge-trap transistor (CTT). It is similar to that of a regular metal oxide silicon (MOS) transistor, except for an additional dielectric layer (charge trapping layer (CTL)) between the blocking layer and the tunneling layer [5]. The dielectric layer between the CTL and the gate is called the blocking layer. The blocking layer is used underneath the Poly-Si capping layer such that the blocking oxide acts as a diffusion barrier to oxygen. The dielectric closest to the substrate is called the tunneling layer or buffer or SI. The name originates from the working principle that the write operation injects electrons through this thin oxide and the erase operation de-traps electrons back into the substrate from this thin oxide.

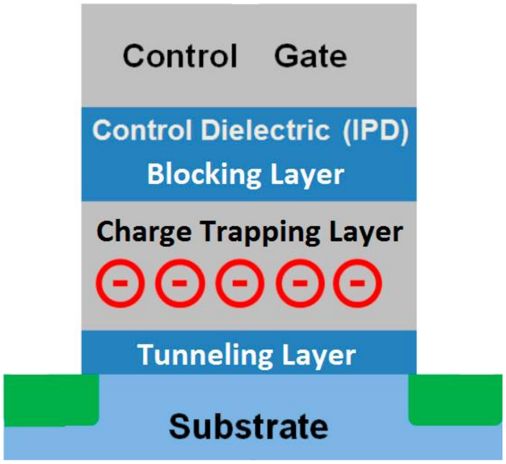


Figure 2.3 Cross-section schematic of a charge trapping memory transistor [7].

As the name indicates, the charge-trapping or HfO_2 layer serves as the charge storage layer. The write/erase operation is determined by charges insertion and removal from the CTL. The write/erase speed is usually defined as threshold voltage variation with respect to time or write/read operation cycle. For example, in the write operation, a high positive voltage is applied to the gate, causing electrons to be injected from the substrate into the CTL and leading to a positive shift of threshold voltage. In the erase operation, a high negative voltage is applied to the gate in order to cause electrons escaping from the CTL into the substrate, resulting in a negative shift of threshold voltage. To avoid destruction and short lifetime, a low operating voltage with a short pulse-width is desirable for write/erase operation. The thickness of the CTL influences the speed of write/erase and the magnitude of read current. Low defect density and long mean time to failure, together with charge retention capability, are important reliability issues of the CTL [7] [11].

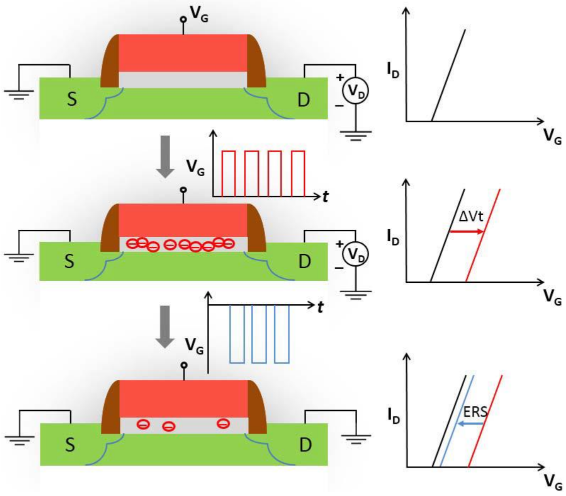


Fig. 2.4 Threshold voltage variation while write and erase operations [5].

2.4 HfO_2 AS CHARGE TRAP LAYER

SiO_2 gate dielectric failed to scale beyond 2nm due to excessive gate leakage, gate dielectric breakdown and reliability issues [12]. Therefore, it became necessary to replace SiO_2 with higher dielectric permittivity (high-k or HK) material in order to continue the EOT (equivalent oxide thickness) scaling. Considering many alternatives, HfO_2 turned out to be the most promising material for CMOS technology. This section discusses the advantages of using HfO_2 as the CTL and its potential for memory application. It is found that electrons were trapped both at the interfaces and in the bulk of HfO_2, in contrast to Si-based materials, where electrons are trapped mainly at the interfaces. The main reason of bulk traps in HfO_2 is oxygen vacancies [1] [13]. From the viewpoint of EOT scaling, HfO_2 maintains electron trap capability even when EOT is less than 1 nm, not possible with Si-based materials [9]. Even at elevated temperature, the de-trapping of electrons from HfO_2 is gradual due to bulk traps and a wider distribution of trap levels, leading to longer lifetime.

In addition, other factors that help HfO_2 to emerge as a leading candidate to replace Silica gate dielectric are its κ value (20–25), thermal stability, large heat of formation (271kcal/mol, higher than that of Silica: 218kcal/mol), large band gap (5.5–6.0eV) and high barrier height (1.3eV) greatly reducing electron tunneling and leakage current [4] [13]. HfO_2-based materials are now widely researched as insulating layer in CMOS technology by overcoming the problems and routinely used as 10’s nm CMOS processes.

2.5 SCALABILITY IN NVM

According to Moore’s law, scaling the gate oxide and the gate length is the most effective way to improve performance and reduce costs. Gordon Moore suggested that for every two years, the number of devices per unit area will be doubled. This trend has proved to be correct and scaling continued steadily through technology nodes in microns on to nanometer-scale reaching 10nm, possibly angstroms-scaling in the near future [4] [14].

The size of the transistors decreases every time the device is scaled due to which the size of the interconnects have got smaller and this has reduced the path length for electrons to travel. Some other benefits of device scaling are small circuit delays, low power consumption and high speed of device operation [4].

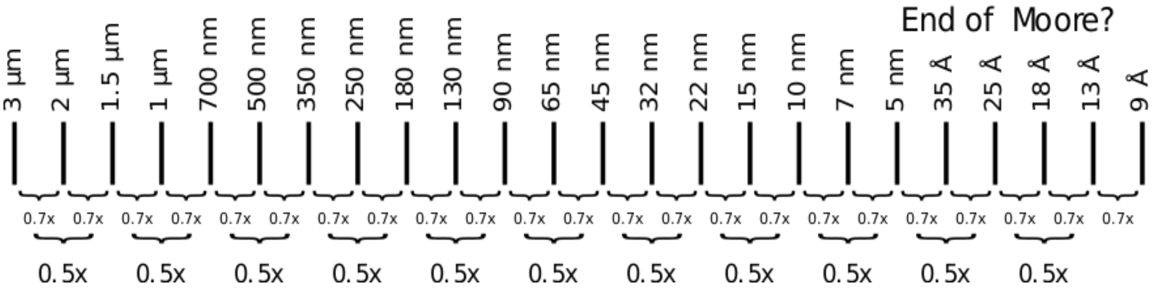


Figure 2.5 The scaling of feature size [Wikipedia].

According to Robert Dennard in 1974, the dielectric thickness must decrease along with the physical dimensions of the device and the gate length which is the parameter to distinguish the technology node [4]. This means that gate dielectric thickness will reach atomic dimensions by continued down-scaling. However, the SiO_2 gate dielectric oxide failed to scale beyond 2nm, the solution is to replace Silica with new gate oxides called 'High-κ oxides' that are physically thicker for scaling down to 45 nm and below nodes. Any material with higher k value, good electron trapping capabilities and retention characteristics can be used as long as the equivalent oxide thickness (EOT) is being scaled. EOT is defined as the thickness of Silica that would hold the capacitance which is equal to the device being measured by: [4]

$$t_{OX} = EOT = t_{HiK}\frac{3.9}{k}$$

where, 3.9 is the relative permittivity constant of Silica.

This work uses HfO_2 as the charge trapping layer, we make a comparison between HfO_2 with Si-based material which is mostly used for nodes above 45nm nodes. On comparison, ΔVsat and oxygen vacancies existence due to their ionic nature are not affected by the thinning of the gate oxide, even when the EOT is less than 2nm in case of HfO_2 as CTL [9]. Whereas, in Si-based materials like silica and SiN, it became increasingly difficult to achieve reliable gate oxide thinning below 1-2 nm, leading to uninvited gate leakage currents, stress induced leakage currents and gate oxide unreliability [13] [4]. Due to these ill effects, stand by power consumption had risen disturbingly while it remained near constant in higher *k* value materials.

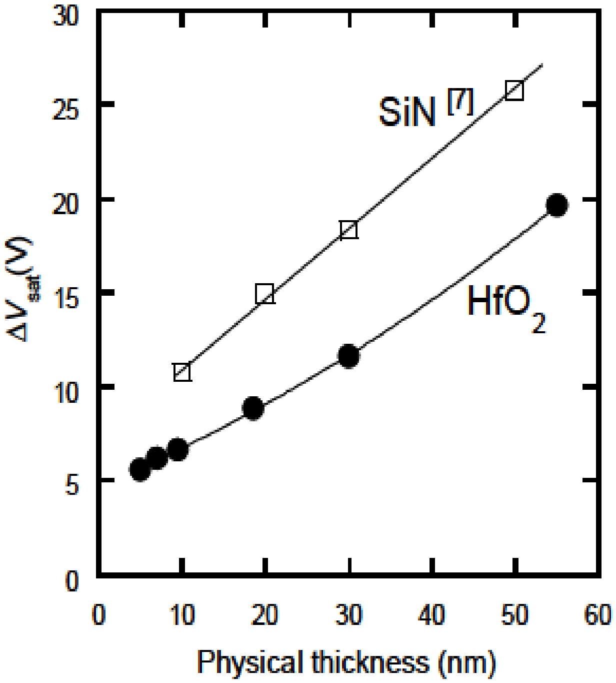


Figure 2.6 Dependence of V_{sat} on physical thickness (SiN and HfO$_2$) [9].

In this chapter, we discussed the security attacks on memory. Then, we propose the solution for these attacks as EEPROM memory that employs the charge-trap transistor (CTT) to store data. The structure and the working principle of CTT is explained. Various factors are discussed for using HfO$_2$ as charge trapping layer. The scalability in non-volatile memory and dielectric oxide are discussed. The work takes advantage of ready HfO$_2$ as dielectric in 42nm CMOS.

CHAPTER III

EEPROM WRAPPER ARCHITECTURE

3.1 INTRODUCTION

The interface (shown in figure 3.1) consists of three input registers, a finite state machine (FSM) controller, counters, clock divider, necessary combinational logic and command user interface (CUI) logic. The interface connects the EEPROM and CPU with four bus signal groups, namely data bus, R/W signal, address and clock. The R/W signal indicates CPU read or write and the address specifies input register. ECC functional block is reserved for insertion into future versions [6]. Transporting data in parallel through the interface and moving memory from off-chip to on-chip greatly prevents power analysis based side-channel attacks by decreasing signal power by 4 or 5 order of magnitude while decreasing SNR by at least 2 order of magnitude.

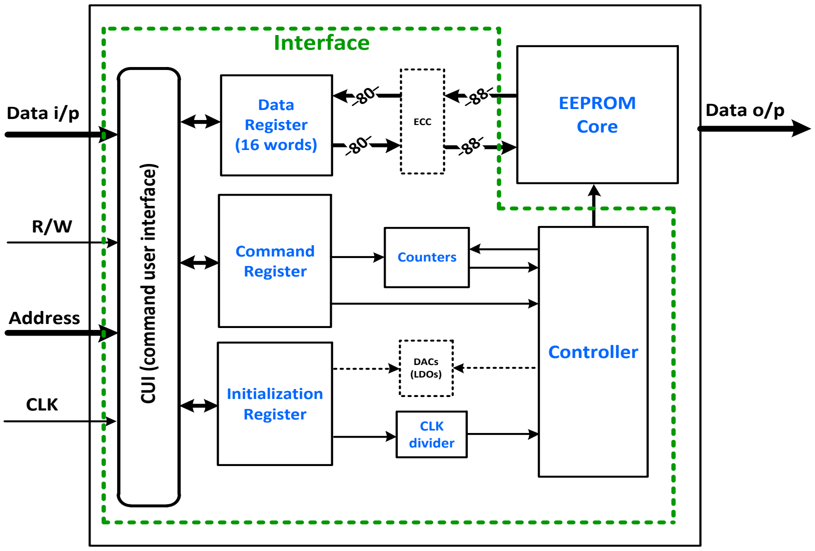


Figure 3.1 EEPROM and its interface with CPU.

3.2 INPUT REGISTERS

There are three input registers, *initialization register, command register and data register*. All the registers are 80 bits in length.

3.2.1 Initialization register

Initialization register has 80 bits consisting of the CPU clock dividing value and programming voltage values. Bit allocation of the initialization register is shown in figure 3.2. The voltage programming bits are reserved for future on-chip or off-chip DAC control of programming voltages. Programming voltages are supplied by the test PCB in the present version.

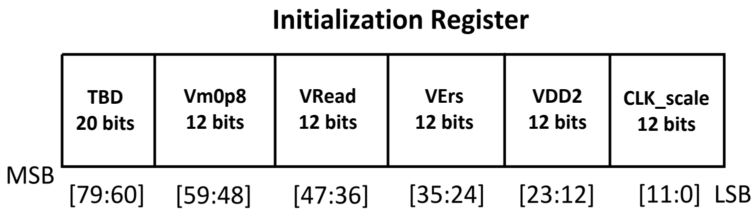
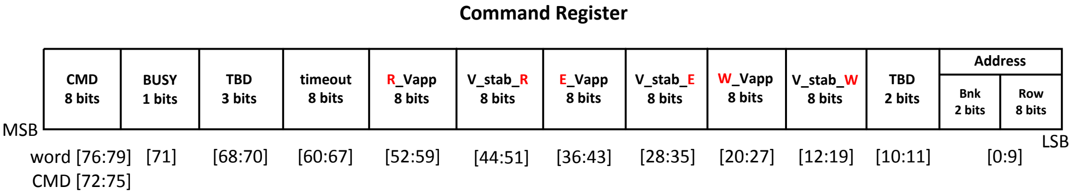


Figure 3.2 Initialization register bit allocation.

3.2.2 Command register

Command register has 80 bits consisting of the EEPROM address, counter/timer values of voltage stabilization time and voltage application time for write, read, and erase operations respectively, time out value, a **BUSY** bit and command bits. Figure 3.3 shows the bit allocation of command register.

The **BUSY** bit is an access control bit in the CUI between CPU and EEPROM. The **BUSY** bit prohibits the CPU from writing the data register until a command execution is completed by the EEPROM. The address bits **CMD[0:9]** include word and bank address pointer to the EEPPROM location to be written/read/erased. Bits **CMD[10:11]** are reserved for future use. Voltage stabilization time bits, **CMD[12:19]**, **[28:35]** and **[44:51]**, are time delays ensuring operation voltages are stable at the supply pins before applying or changing voltage to the EEPROM cell by control signals. Voltage application time bits, **CMD[20:27]**, **[36:43]** and **[52:59]**, sets the time period that the programming (W/R/E) voltage is to be applied. The timeout bits, **CMD[60:67]**, are provided for high voltage supply quenching to avoid excess voltage as the state transitions back to the idle mode. Bits **[68:70]** are reserved for future use. **CMD[72:79]** 8 bits, specify the memory operation to be carried out. **CMD[76:79]** 4 bits, specifies a block (16 words) when equal to xXf and a single word when equal to xX0. The type of operation is specified by **CMD[72:75]**. Write (**WR**) operation is x1X, Erase (**ER**) operation is x2X and Read (**RD**) operation is x3X, respectively. All other combinations of **CMD[72:79]** are invalid for this design and reserved for future use.



Command Register

											Address	
CMD 8 bits	BUSY 1 bits	TBD 3 bits	timeout 8 bits	R_Vapp 8 bits	V_stab_R 8 bits	E_Vapp 8 bits	V_stab_E 8 bits	W_Vapp 8 bits	V_stab_W 8 bits	TBD 2 bits	Bnk 2 bits	Row 8 bits
word [76:79] CMD [72:75]	[71]	[68:70]	[60:67]	[52:59]	[44:51]	[36:43]	[28:35]	[20:27]	[12:19]	[10:11]	[0:9]	

Figure 3.3 Command register bit allocation.

3.2.3 Data register

Data register consists of 16 words with a width of 80 bits used solely for temporary data storage (buffering) between CPU and EEPROM. These data registers are controlled by the **BUSY** bit. If the **BUSY** bit is 0, then CPU has the access to the data registers. CPU decides whether to write or read the data registers. Whereas, if the **BUSY** bit is 1, then EEPROM has access to the data registers and can perform write or read operation on them. The data register serves as an intermediate buffer between CPU and EEPROM to minimize the overhead time since EEPROM operates at a much lower speed than the CPU.

3.3 COMMAND USER INTERFACE

Command User Interface, (CUI) is the logical block that controls the data traffic between CPU, input registers and EEPROM core. Figure 3.4 presents the CUI circuit diagram. This unit consists of an initialization register, command register, sixteen (a block of 16 words) 80-bit data registers, a 4-bit word counter, and other control logic. Registers can be accessed by applying the correct CPU address which uses the two least significant bits (LSB) of the CPU address, **CPU_addr[0:1]**. The **CPURW** bit indicates that CPU writes to the registers when it is logic 1 and reads from registers when it is logic 0. Both CPU and EEPROM core access the 16-word data register in a mutually exclusive manner, depending on the status of **BUSY** bit. A clear **BUSY** bit, logic 0, allows CPU to access the data register. CPU write or read depends on the **CPURW** bit status. When the **BUSY** is true, logic 1, the EEPROM has access to the data register. In this manner CPU and EEPROM can never access data registers simultaneously since the CPU sets the **BUSY** bit passing control while the wrapper clears the **BUSY** bit passing control back. *The CPU may read initialization and command registers independent of the status of the **BUSY** bit.*

The 4-bit word counter tracks operation progress through the 16 words data register. For a block (16 words) of data, the word counter is loaded with value xXf via the command register bits **CMD[76:79]**. For single word data, the word counter is loaded with value xX0. Granting CPU access to the data register enables the word counter. The word counter is decremented via the CPU clock as execution progresses through the data register count. The word count is decremented during EEPROM register access, for each data register read by the EEPROM during the write operation. The word counter is decremented only when signal **tWR_done** or **tRD_done** are high which states that one cycle of write or read operation is completed. The data register latches data when **tRD_done** is true indicating the content read from EEPROM memory is written back into the data register. For both CPU and EEPROM access, when either a block (16 words) or a word access is finished, the overflow bit, **Cout,** of the word counter goes high indicating the completion of operation. Relevant state(s) within the wrapper state machine are relative to the **Cout** signal.

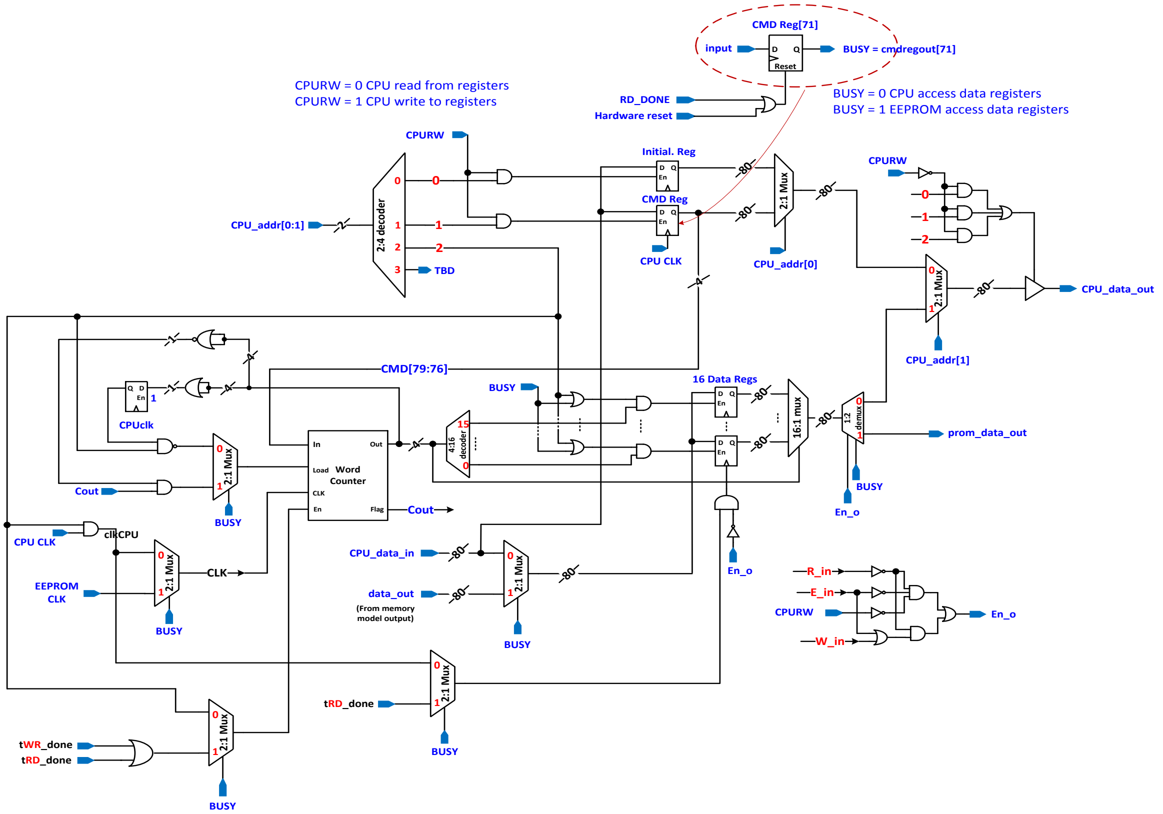


Figure 3.4 CUI (command user interface) schematic.

3.4 CONTROLLER (FINITE STATE MACHINE)

Figure 3.5 summarizes EEPROM wrapper operation using a state diagram. This design uses separate counters/timers for each state of 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 during the “Write starts” state or “Read Starts” state or reloaded for next word in the “W_done” state. The voltage application timer starts in the “W_Vapp time” state. In “W_done” state, check for **Cout**=’1’ and **t_out**=’0’ which depicts the completion of an operation. **t_out** signal is the timeout indicator. It is high when timeout counter is done counting.

When **Cout**='0' and **t_out**='0', control should return to the "W_Vapp time" state that means there are still words to write. Timeout timer starts only when **Cout** equals 1 irrespective of being in the "W_done" state indicating there are no more words in data register to be written. After the write operation is completed, the CUI enters the read operation automatically to read out the words as written or as erased and stores them back into the data registers for CPU retrieval and confirm the data for accuracy.

Erase voltage stabilization timer is loaded in the "Erase Starts" state and enabled in the "VErs V_Stab" state. The erase voltage application timer is loaded in "Erase Starts" state or "Read Starts" state and enabled in the "E_Vapp time" state. Since only a block (16 words) is erased at eraser, timeout counter starts on the first "E_done" state occurrence, the word counter is loaded with value "1" and sufficient voltage application time is completed so as to erase all 16 words simultaneously. After erase time out, i.e. **t_out** is true, the read operation starts automatically to confirm erasure. This design only does a block erase. *The block access is restricted to addresses that have a 16-bit boundary available within the same bank. In other words, this design does not support block operations that overlap between two banks i.e., all the 16 addresses of the block must be in the same bank.*

The read voltage stabilization timer is loaded in the "Read Starts" state and enabled in the "Vread V_Stab" state. The read voltage application timer is loaded in "Read Starts" state or reloaded for next word in the "R_done" state and enabled in the "R_Vapp time" state. After the completion of read operation, the control returns to "idle" state and waits until it receives a new operation command from the CPU.

The write operation and erase operation share the *Time out counter* and the time out timer is loaded in either the "Write starts" state or "Erase starts" state. It starts the time out delay only if the control is in the "E_done" state or "W_done" state and **Cout** is true.

This controller also generates control signals **W_in**, **E_in**, **R_in**, **WR**, **ER**, **RD** and **RD_DONE** that are required for processing few other signals in the VHDL code. Signals **W_in**, **E_in**, and **R_in** stay high from the time the control enters the “W/E/R_Vapp” state till the “W/E/R_done” state is active of the corresponding operation. Signal **WR/ER/RD** is true only during the “W/E/R_Vapp” state of the corresponding operation. **RD_DONE** is generated when the read operation is completed and the control is returned to “idle” state and used to reset **BUSY** bit so that CUI access is returned to CPU.

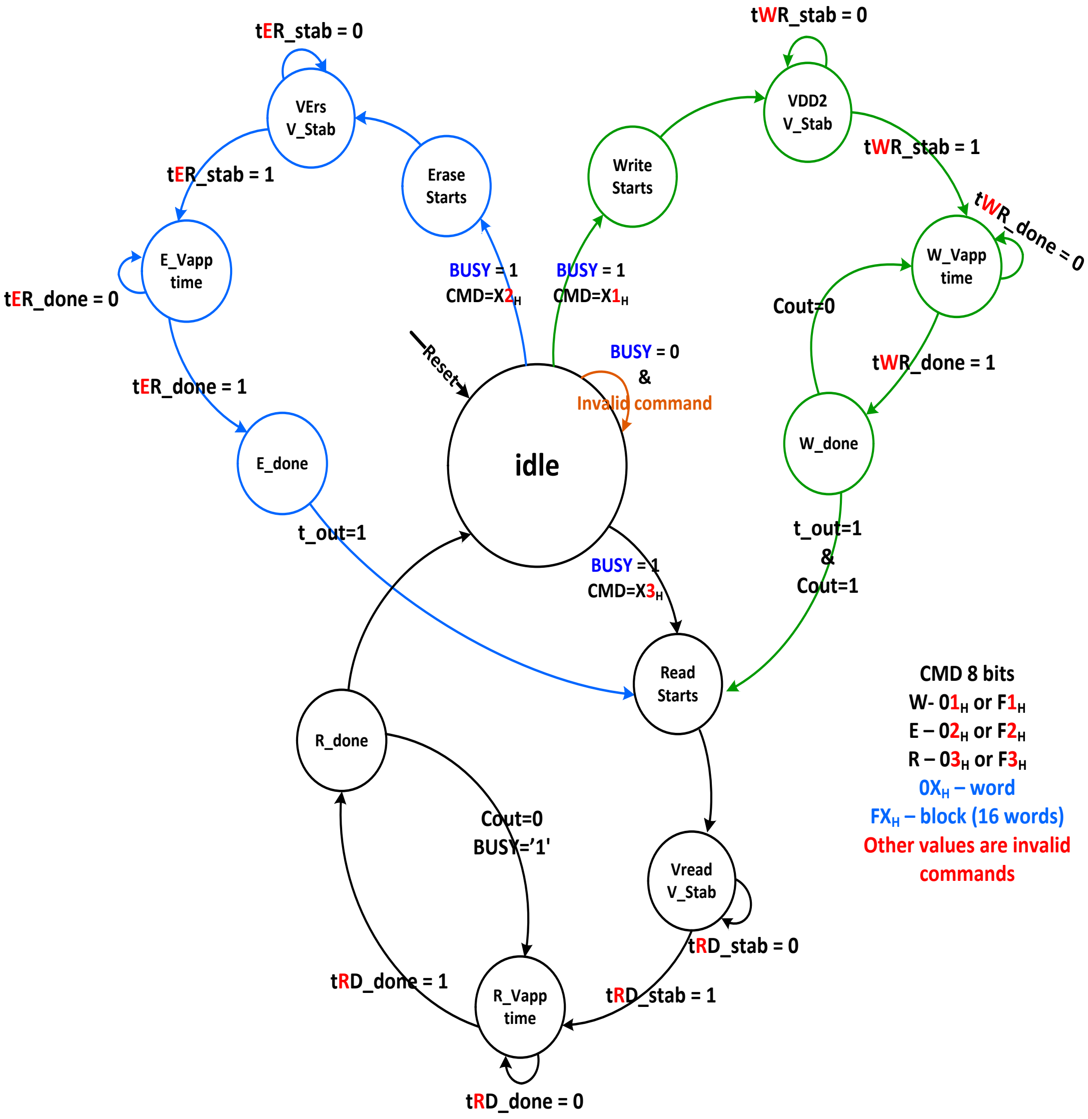


Figure 3.5 State diagram of the controller

3.5 WRAPPER SIGNALS AND SCHEMATIC

Figure 3.6 shows the signals between the CPU, wrapper/interface and EEPROM core that are crucial for the understanding of the wrapper. CPU provides input data (CPU_data_in), clock for synthesis (CPU Clk), Write/Read indicator (CPURW) and address (CPU_addr) to the wrapper. This is given as the test bench while executing the VHDL code. Then, the wrapper interprets the input using the VHDL logic code, where the data has to be read/written and performs the corresponding operation on the selected bank using the controls signals, CS_Bk, CSbar_Bk, N_Bk, Nbar_Bk, TL_Bk. SE signal comes into play only if a Read operation is performed. The circuit in the dotted box of figure 3.6 represents the row select (RS) and it is repeated 256 times for all the words. After the completion of an issued operation, the EEPROM memory model provides processed data as an output to the wrapper (data_out_bk). Finally, the wrapper delivers the final data output (CPU_data_out) to the CPU for retrieval and confirmation that the data processed (written or erased) is valid and accurate.

Coming to the dependency on the circuit, the written/erased data accuracy depends on their voltage application that can overcome the threshold voltage of the gate. The applied voltage has to overcome the offset voltage and trigger the sense amplifier during the read operation.

Figure 3.7 shows the full wrapper schematic. It gives the block level implementation of interface architecture for control and data communication between the CPU and the EEPROM core. Each bit operation of input registers is shown. All the components of command user interface are elaborated and their operation is explained at signal level.

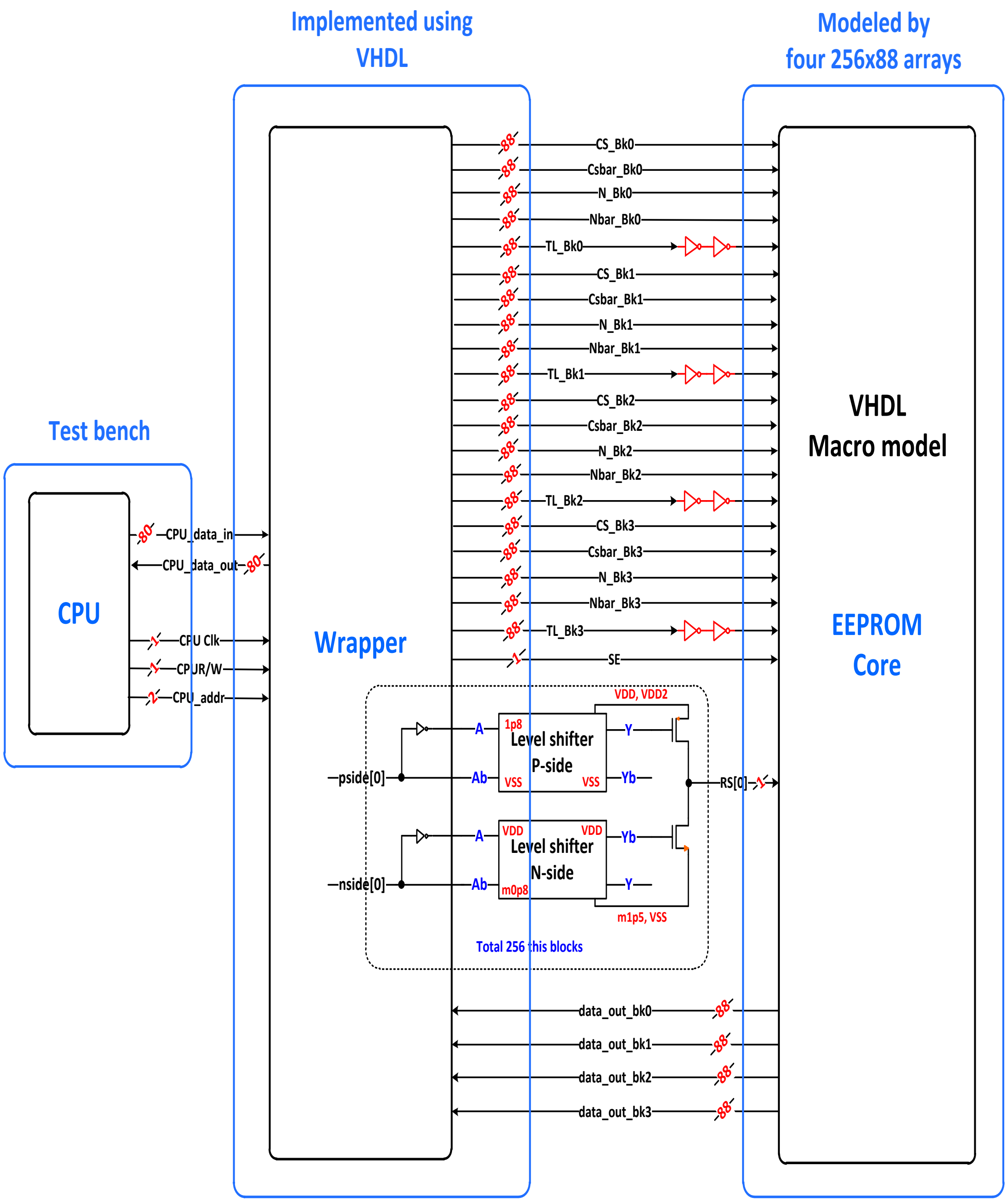


Figure 3.6 CPU, wrapper and EEPROM interconnect signals

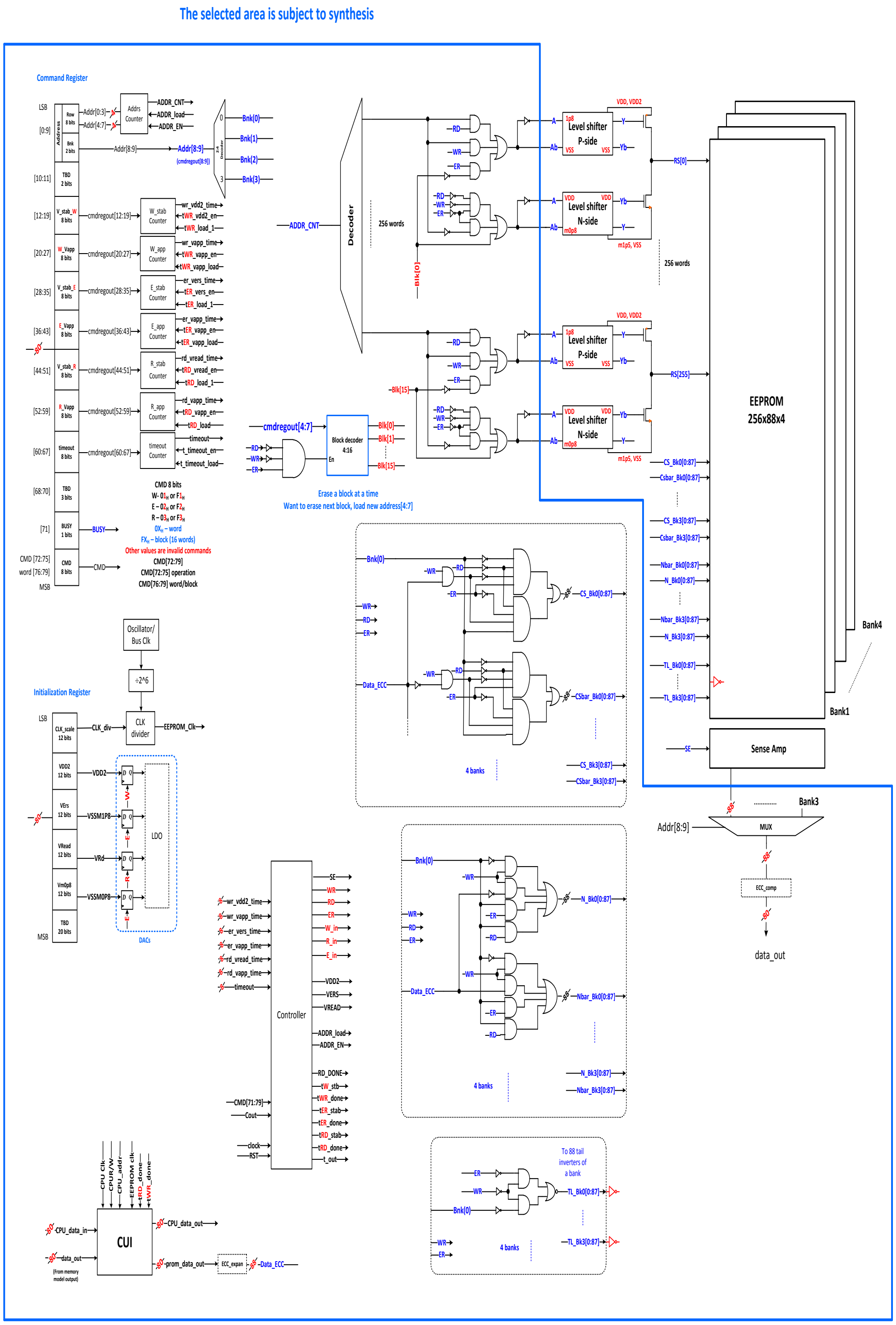


Figure 3.7 Wrapper schematic.

CHAPTER IV

FUNCTIONALITY OF INTERFACE USING VHDL AS MEDIUM

4.1 WRAPPER OPERATION

4.1.1 Write

The write operation begins by the CPU observing the BUSY bit false, loading the initialization, command, and data registers and setting the **CPURW** bit true. **CPU_addr** ‘00’ is the initialization register, ‘01’ the command register and ‘10’ the data register. **CPU_addr** ‘11’ is reserved for future use. When loading the command register for the first time, the **BUSY** bit (**CMD[71]**) should be loaded with 0 allowing the CPU to have access to the wrapper registers. After the data register is loaded, the command register is reloaded to set the **BUSY bit,** passing control to the EEPROM wrapper state machine. This will start the designated EEPROM operation set by the command bits **CMD[72:79]**.

Example CPU commands for write:

```
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”01002050202020202000”; -- write one word, BUSY bit is 0, CPU access data register.

wait for 10 ns;


CPU_addr <= “10”; -- data register

CPU_data_in <= x”a0000000000000000000”;

wait for 10 ns;

CPU_addr <= “01”; --reload command register

CPU_data_in <= x”01802050202020202000”; --BUSY bit is set to 1, EEPROM operation starts.
```

Command register bits **CMD[72:79]** should be x10 or x1f indicating write one word or a block (i.e.16) of words respectively. *The EEPROM wrapper returns to the idle state and access to data register is returned to the CPU when an invalid command is issued.* Command register address bit **CMD[0:9]** is decoded to allow access to the target address and bank. When writing a block of words, the initial address is decoded. The word counter in CUI is loaded with xXf. The EEPROM address counter increments to the next addresses sequentially after each word written. The word counter is decremented pointing to the next word being moved to memory from the data register. This process repeats until all 16 words (one block) are written to memory. *This design does not support block operations that overlap between two banks. This means block access is consecutive from the address of the first memory word only on 16 bit boundaries. The starting block address must be an integer multiple of 16 on a 16-word boundary*. After the write operation is complete, the read operation automatically starts to confirm the write [6]. The stored words are written back to the data register for CPU to retrieve and confirm. After the read operation is completed, data register (or CUI) access is returned to CPU by the FSM which clears the **BUSY** bit. The interface control then enters idle state. The CPU may then retrieve the data from the data register or disregard it by over writing the data register with new values.

The EEPROM control signals to each bank for the write operation are **CS_Bk0[0:87]**, **CSbar_Bk0[0:87]**, **N_Bk0[0:87]**, **Nbar_Bk0[0:87]**, **TL_Bk0[0:87]**. These signals are generated for each of the four banks by the decoded bank address from command register bits **CMD[8:9]**. This design has preserved 8 bits for ECC. Therefore, the 80-bit data will be appended by 8 bits to the MSB side after coming out from CUI to make 88-bit ECC data in the memory core. All the EEPROM control signals are 88 bits. The control signals **CS_Bk0[0:87]**, **CSbar_Bk0[0:87]**, **N_Bk0[0:87]**, **Nbar_Bk0[0:87]**, **TL_Bk0[0:87]** are generated from the signals **Bnk**, **WR**, **RD**, **ER**, **Data_ECC** which are generated from the controller. Section 4 discusses control signal generation in detail.

4.1.2 Erase

The erase operation simultaneously clears a block of words [8]. The erase operation begins when the CPU loads the command register **CMD[72:79]** bits with x2F, **BUSY** bit 0, correct timer values and address. The bits **CMD[76:79]** needs to be loaded with xXf for the word counter to enable the 16 words to be read after erase operation. The **BUSY** bit is set via command register starting a new block erase operation. Other **CMD** bits remain the same from previous load. It is necessary for the two separate load operations to the command register for setting up the word counter and **BUSY** bit. The block to be erased is determined by decoding the address bits **CMD[4:7]**. The **BUSY** bit is cleared after the read operation is completed and the interface control enters idle state. To erase an additional block, CPU must initiate via the command register with block starting address and erase command. See section 4 for further details regarding the erase operation.

Example CPU commands for erase:

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”f2002050202020202000”; -- erase command, BUSY bit is 0, **CMD[76:79]** loaded with f.

wait for 10 ns;

CPU_addr <= “01”; -- reload command register

CPU_data_in <= x”f2802050202020202000”; --BUSY bit is set to 1, EEPROM operation starts.

4.1.3 Read

The read operation begins by the CPU loading the command register **CMD[72:79]** bits with x30 or x3f indicating a block read or a single word read respectively, while the **BUSY** bit is clear. In addition, the correct timer values and EEPROM address must be loaded. Again, the BUSY bit is set while other bit values are maintained identical to the first load. The command bits **CMD[0:9]** are decoded to allow access to targeted EEPROM address and bank. The word read result from each memory address are written to the data register for CPU retrieval. When reading a block of words, the initial memory address is decoded, placed in the address counter, and incremented after each word read. After the read operation is completed, the **BUSY** bit is cleared returning data register access back to CPU and the interface control enters the idle state. See section 4 for further details regarding the read operation.

Example CPU commands for read:

```
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”f3002050202020202000”;  -- 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”f3802050202020202000”; -- BUSY bit is set to 1, EEPROM operation starts.
```

4.2 EEPROM STORAGE CELL

A Charge Trap Transistor (CTT) traps charge in the high-k dielectric layer when a high positive gate voltage, nominally 2V, is applied [1] [5] [6]. Whereas, it de-traps charge when a negative voltage, nominally -2V, is applied [1] [5] [6]. Each memory cell consists of a differential NMOS transistor pair, T1 and T2, whose gates are tied together forming a word line as shown in figure 1 [6] [10] [11]. Write operation is conducted by applying a high gate voltage along with input data so that carriers trap into the dielectric of one side of the differential pair resulting in a threshold difference [6] [11]. This threshold difference results in a current difference in Bit line and Bit line bar which is sensed by Sense Amplifier (SA) for a logic output. Erase operation is conducted by applying a negative gate voltage that de-traps the carriers from high-k dielectric layer [6] [7]. After carriers are removed from the dielectric layer, the threshold voltage of the programmed transistor returns to a value close to its native state. The erase residual must be negligible compared to the programming shift [1].

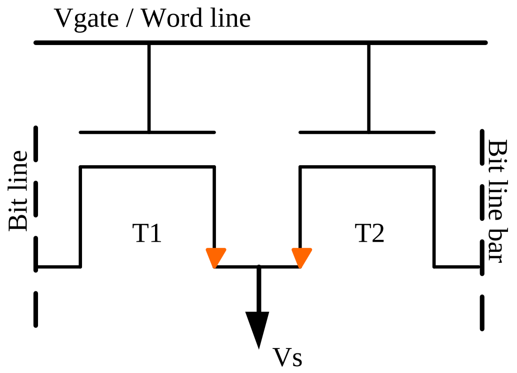


Figure 4.1 CTT based storage cell of the secure EEPROM [6].

4.3 EEPROM CORE CONTROL SIGNALS

The control signals of EEPROM core can be classified into two groups for further discussion. One group of signals controls the transistors of each column to assist cell operation. This 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]** for a bank. *Note there are two inversions between TL signal and the storage transistor terminal. When designing the logic, the two inversions should be considered.* The **Bkn** where 'n' indicates the bank number. For four banks, the names are; **Bk0**, **Bk1**, **Bk2**, **Bk3**. The generation of these signals depend on decoded bank select signal **Bnk**, **WR**, **RD**, **ER**, and **CS/CS_bar.** Figure 4.2 shows the signal location in the circuit. Table 4.1 shows signal status for different memory operations. *One important point is that the transistor on the side where CS or CS_bar is logic 1 will have charge trapped in the dielectric resulting in increase of threshold [10]. When reading, this side will have smaller current than the non-programming side of the storage pair.*

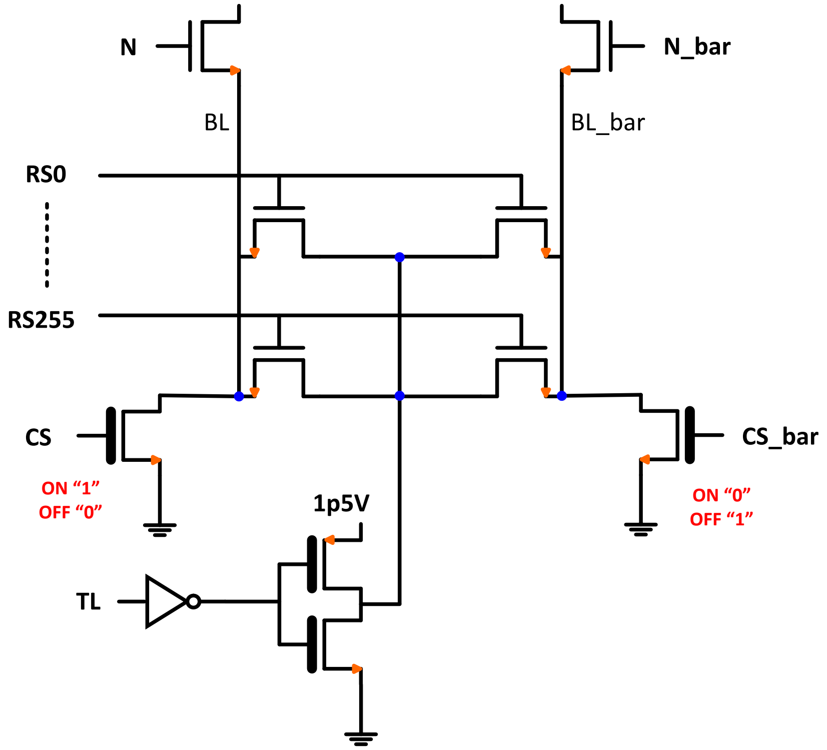


Figure 4.2 EEPROM cell control signals.

The other group of signals include 256 **pside** and **nside** signals that are connected to the level shifter to generate the row select signal and sense enable signal **SE** to enable the sense amplifier [10]. The signal status of **pside** and **nside** are derived from level shifter logic and the RS requirement for each memory operation shown in Table 4.2. Figure 4.3 shows the level shifter logic. Signals **pside** and **nside** are generated based on signals **WR**, **ER**, **RD**, decoded address **addr_decode**, and block select signal **Blk**, which is decoded from address bits **CMD**[4:7]. The red line of synthesis in figure 4.3 shows the boundary of circuit synthesized.

This design has preserved 8 bits for ECC. Therefore, the 80-bit data from data register will be appended by 8 bits to the MSB side after coming out from CUI to make 88-bit ECC data (**Data_ECC**) in the memory core. All the EEPROM control signals are 88 bits.

Table 4.1 Signal status 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

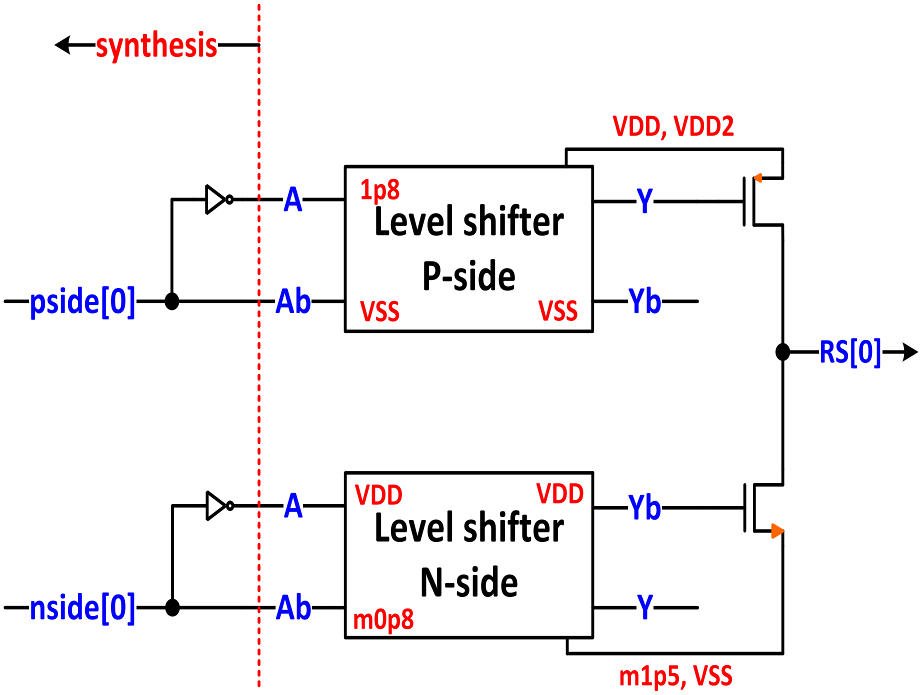


Figure 4.3 Control signal for generating **RS** signal.

Table 4.2 Row select (**RS)** logic

	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

4.4 EEPROM MACRO MODEL

The EEPROM core is modeled by a 256x88 array. Each bank 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]**. Signal conditions are stated in Table 4.1 and coded as following.

```
state_bk0 <=
w_bk0 when
        (
                (CS_bk0 /= CSbar_bk0) and (N_bk0 /= Nbar_bk0) and
                (TL_bk0 = x"0000000000000000000000")
        )else
        e_bk0 when
        (
                (CS_bk0 = x"0000000000000000000000") and
                (CSbar_bk0 = x"0000000000000000000000") and
                (N_bk0 = x"ffffffffffffffffffffff") and
                (Nbar_bk0 = x"ffffffffffffffffffffff") and
                (TL_bk0 = x"0000000000000000000000")
        )else
        r_bk0 when
        (
                (CS_bk0 = x"0000000000000000000000") and
                (CSbar_bk0 = x"0000000000000000000000") and
                (N_bk0 = x"ffffffffffffffffffffff") and
                (Nbar_bk0 = x"ffffffffffffffffffffff") and
                (TL_bk0 = x"ffffffffffffffffffffff")        );
```

The above code is repeated for four banks. When the write operation is performed, **w_bk#** is high. When the erase operation is performed, **e_bk#** is high. When the read operation is performed, **r_bk#** is high. The modeling array is written, read, and erased based on **w_bk#**, **r_bk#**, and **e_bk#** status respectively. The following code shows memory operations based on control signals.

```
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;
```

In this chapter, we discuss how the write/erase/read operation is implemented using VHDL code. The test bench given is shown for each of the operation. The storage cell of our EEPROM is proposed. Therefore, the memory cell structure representing a single bit is designed using various control signals. The values of the control signals are given based on the operation performed. We model the four 256 by 88 EEPROM core arrays based on the control signals as shown in figure 3.6 and the memory operations controlled by these signals.

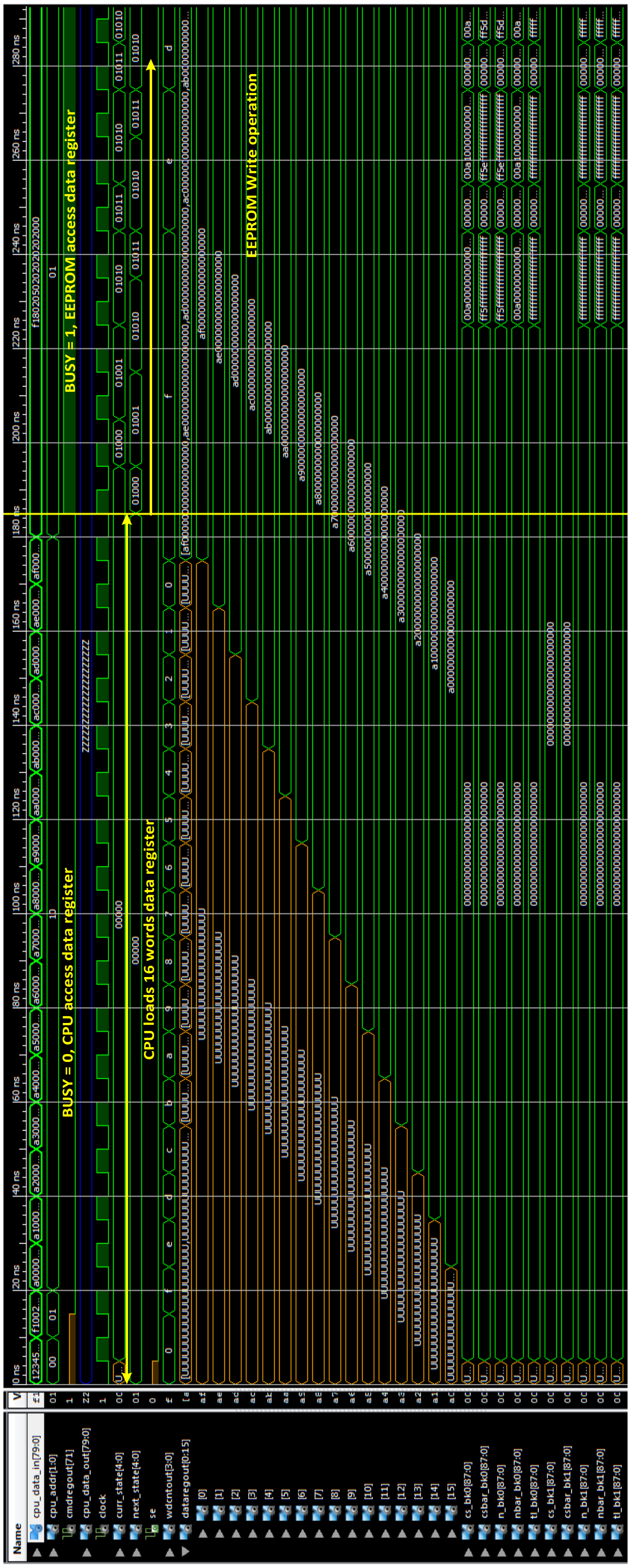
CHAPTER V

RESULTS

5.1 WAVEFORMS OF THE OPERATIONS

The EEPROM interface is logically validated against a VHDL EEPROM memory model that we developed based on charge-trap memory cell operations. The CPU prompts the write/read/erase operation using the command register. The values of command register are given as test bench for the VHDL code. A Xilinx simulator platform is used to validate the VHDL code. The write/read/erase operation of the EEPROM array is performed based on the control signals generated in the interface. The cursor guides the start of each operation in respective waveforms. The waveforms of each operation are divided into phases and each phase signifies a part of respective operation simulation and its corresponding explanation (text box). Figure 5.1 shows the waveforms of the write operation. The CPU writes the data into the 16-word data registers. This is followed by the CUI writing all the 16 words from the data registers to the EEPROM array. After the write operation completion, the CUI reads back the EEPROM memory into the data registers for the CPU retrieval and confirmation. Figure 5.2 shows the waveforms of the erase operation. The CUI erases the block of words from EEPROM array as per the CPU command. Then, followed by the CUI reading back the erased memory block into the data registers for CPU retrieval and confirmation of the erase operation. Figure 5.3 shows the waveforms of the read operation. The CUI, as per the CPU command, reads the 16 words from EEPROM array into the data registers for CPU retrieval and confirmation of the read operation.

Figure 5.1 Waveforms of the write operation.



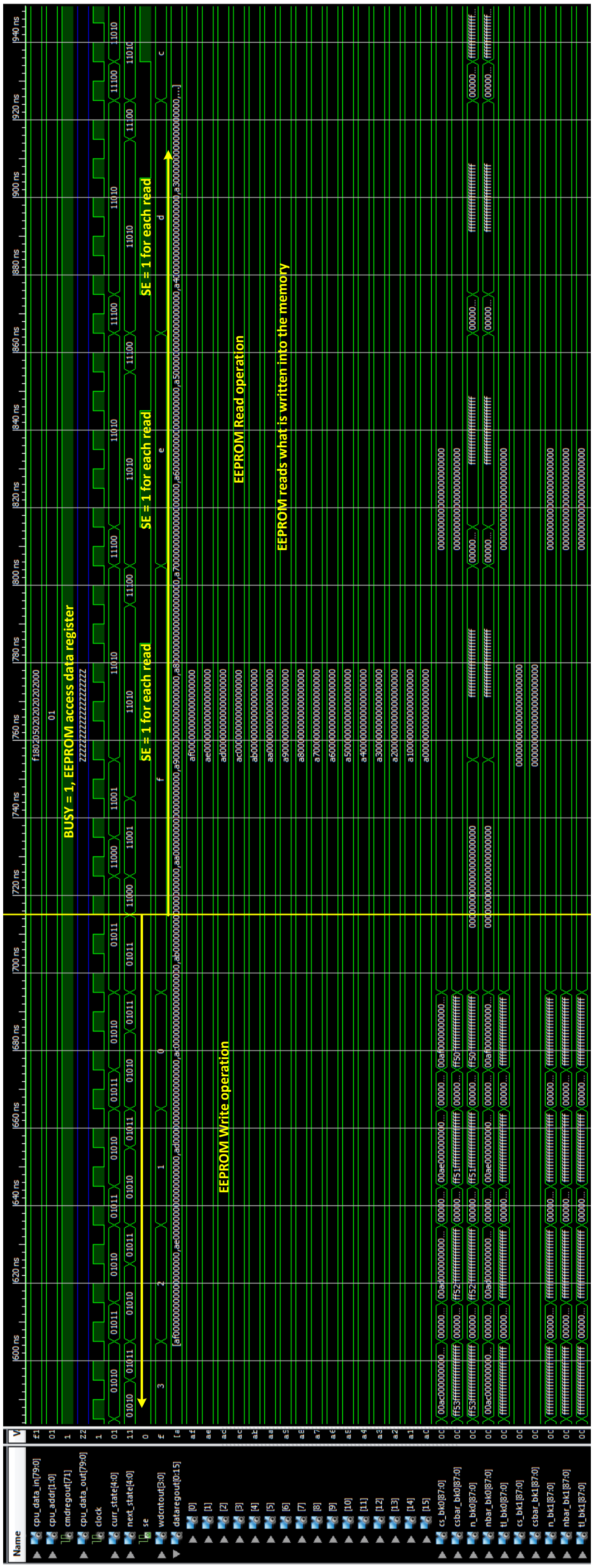
Write a block (16 words)

This waveform shows the transition from CPU writing the data register to EEPROM Write operation.

The BUSY bit is 0 showing CPU accessing data register.

The BUSY bit is set to 1 showing EEPROM has the access to data register.

The staircase waveform shows that the data is written into the data register each clock cycle by the CPU.



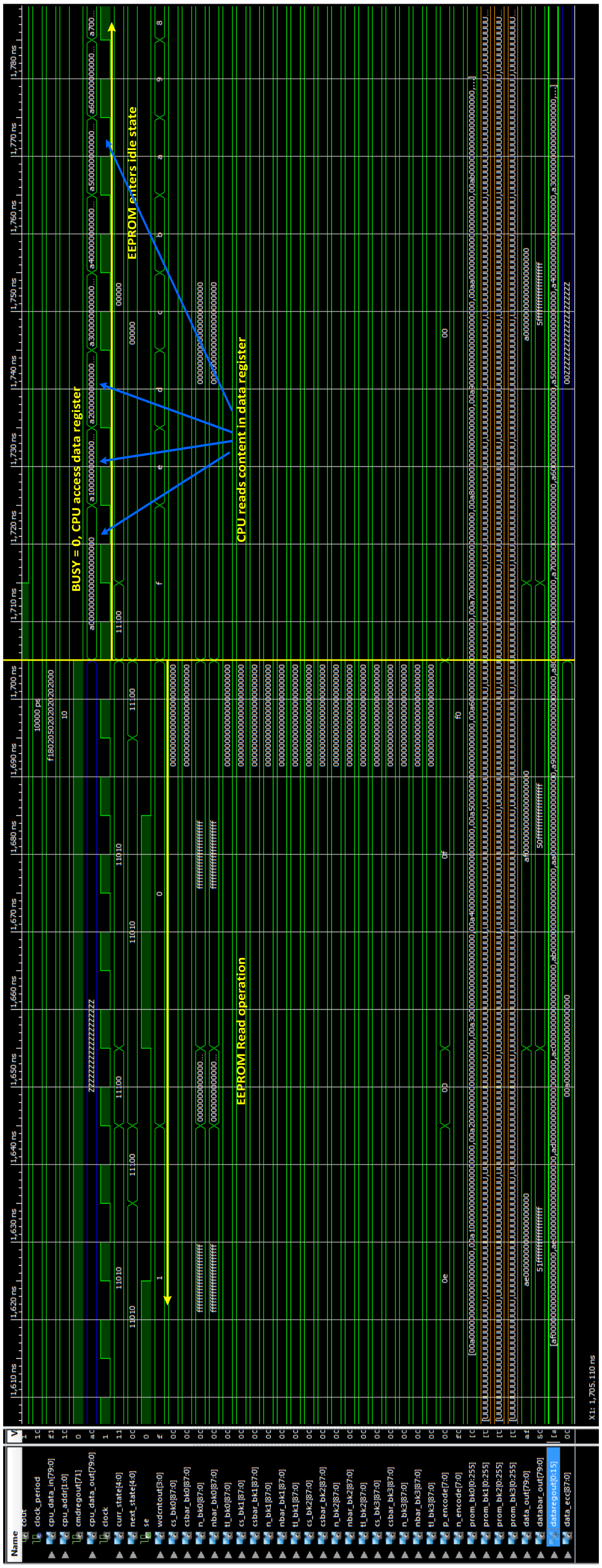
Write a block (16 words)

This waveform shows the transition from EEPROM write operation to EEPROM read operation.

Data register is written with the values that read from EEPROM

BUSY bit is 1 showing EEPROM has access to the data register.

SE, sense amplifier enable signal is high for each read operation.

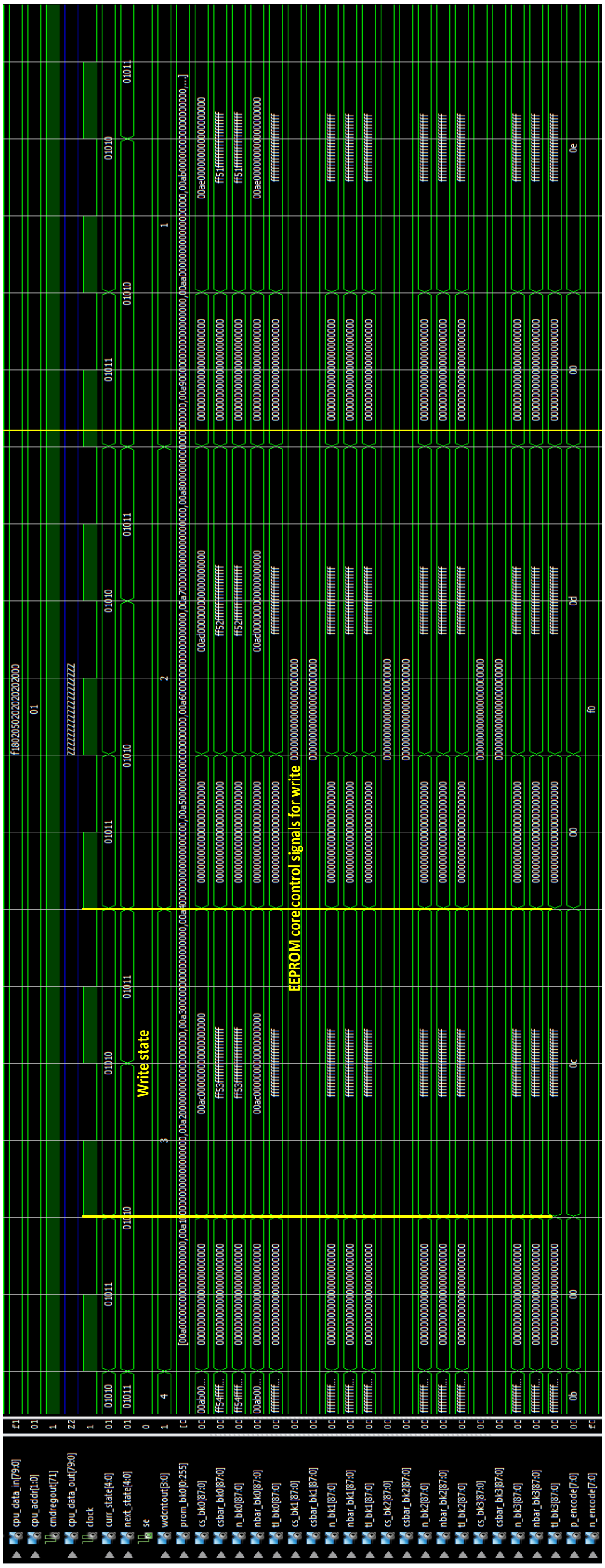

BUSY = 0, CPU access data register
CPU reads content in data register
EEPROM enters idle state
EEPROM Read operation

Write a block (16 words)

These waveforms show the transition from the EEPROM read operation to the CPU data register access.

After EEPROM read is complete, the BUSY bit is cleared showing that the CPU has access to the data register and EEPROM control enters idle state.

After the CPU regaining access to the data register, it reads out the content of the data register for CPU retrieval and confirmation.



This waveform shows the EEPROM core control signal status for write operation.

Write bank

CS_Bk# = Data_ECC

CSbar_Bk# = Data_ECC bar

N_Bk# = Data_ECC bar

Nbar_Bk# = Data_ECC

TL_Bk# = 1 **Non write bank**

CS_Bk# = 0

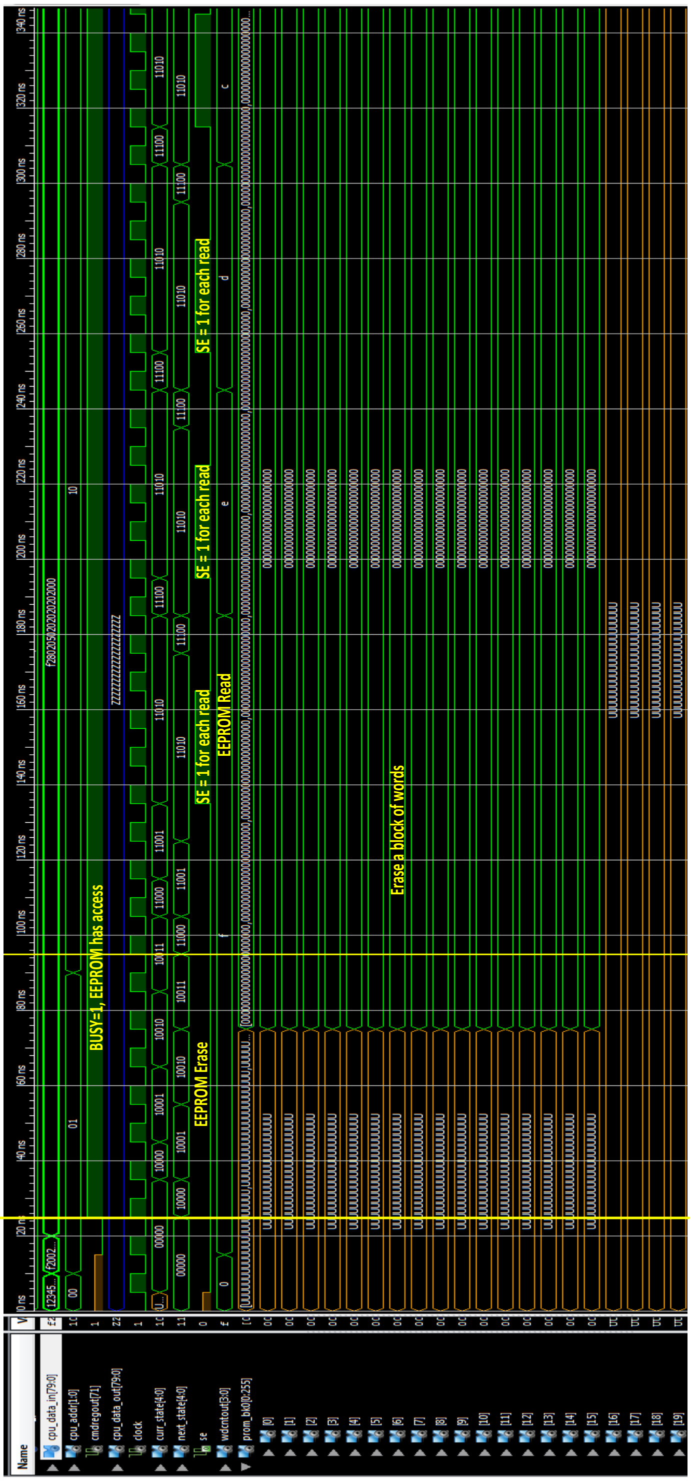
CSbar_Bk# = 0

N_Bk# = 1

Nbar_Bk# = 1

TL_Bk# = 1

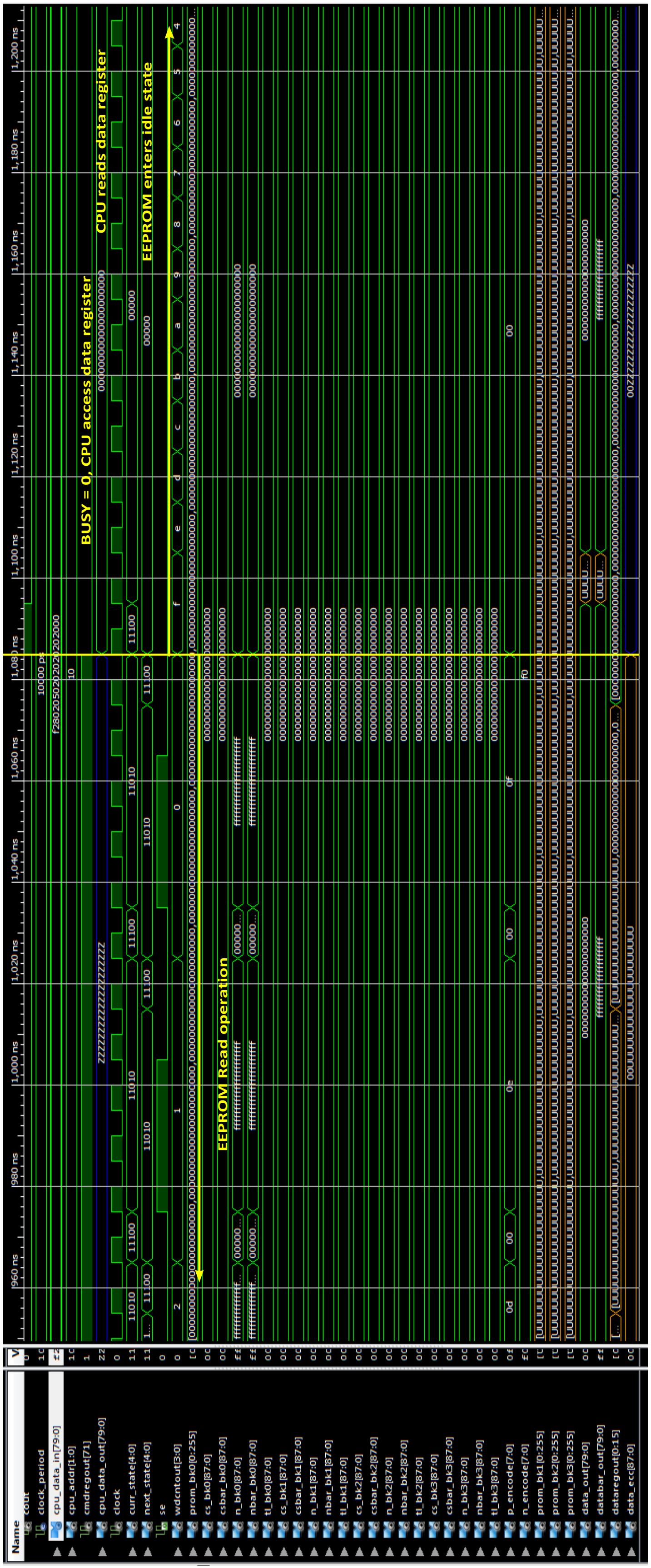
Figure 5.2 Waveforms of the erase operation.



Erase a block

This waveform shows the transition from CPU access of the data register for erase, followed by EEPROM erase to EEPROM read.

In this action, a block (16) of words is erased.



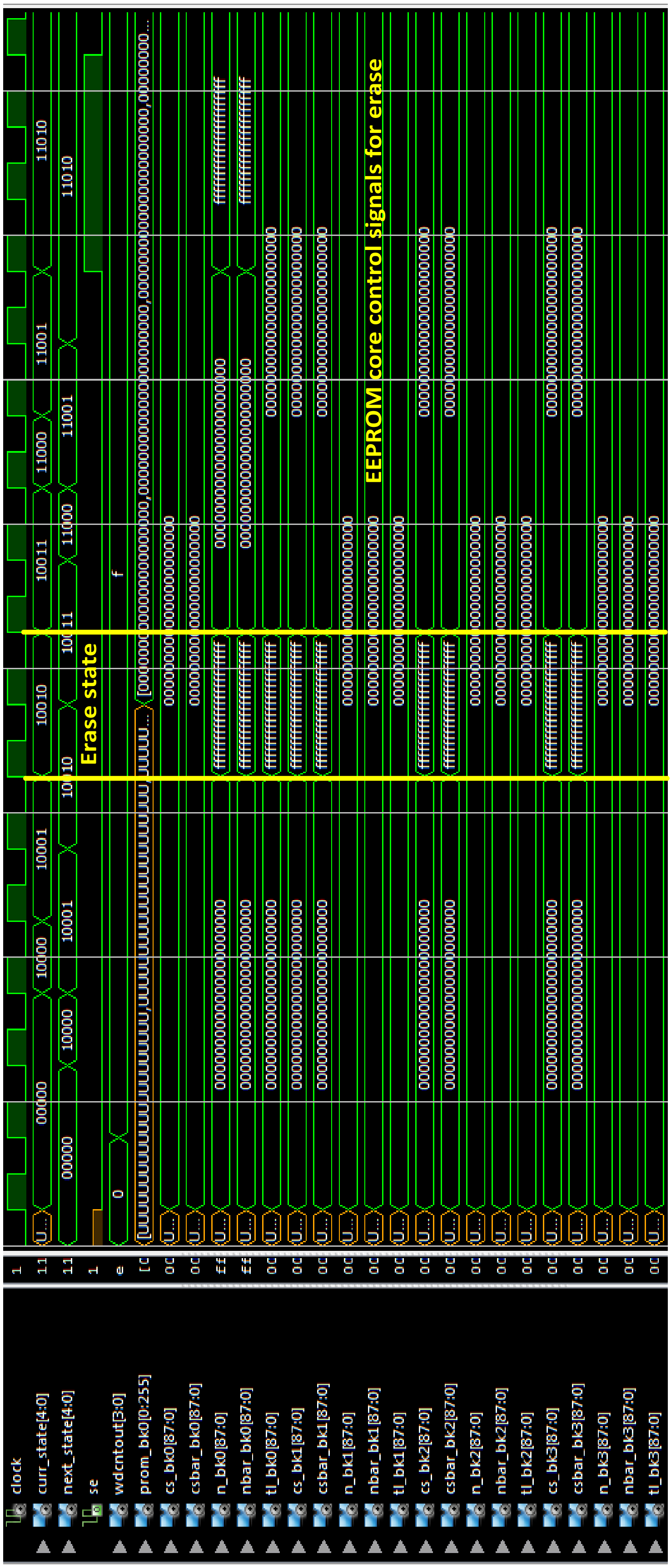
Erase a block

This waveform shows the transition from EEPROM read operation to CPU reobtaining data register access.

After EEPROM read is complete, BUSY bit is cleared showing that the CPU has access to the data register and EEPROM control enters idle state.

After the CPU regaining access to the data register, it reads out the content of the data register for CPU retrieval and confirmation.

For erase, all the content read is zero.



This waveform shows the EEPROM core control signal status for the erase operation.

Erase bank

CS_Bk# = 0

CSbar_Bk# = 0

N_Bk# = 1

Nbar_Bk# = 1

TL_Bk# = 1

Non Erase bank

CS_Bk# = 1

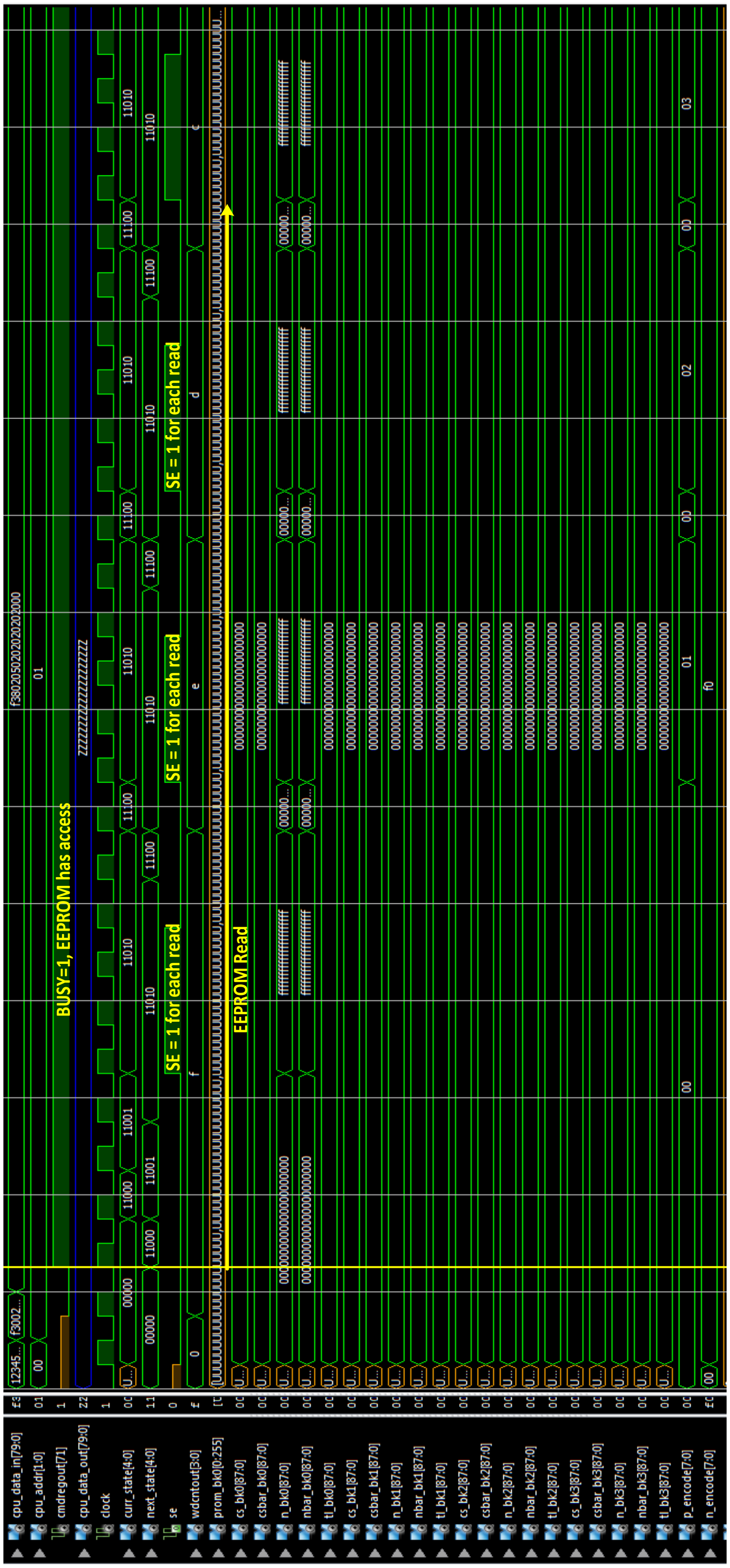
CSbar_Bk# = 1

N_Bk# = 0

Nbar_Bk# = 0

TL_Bk# = 0

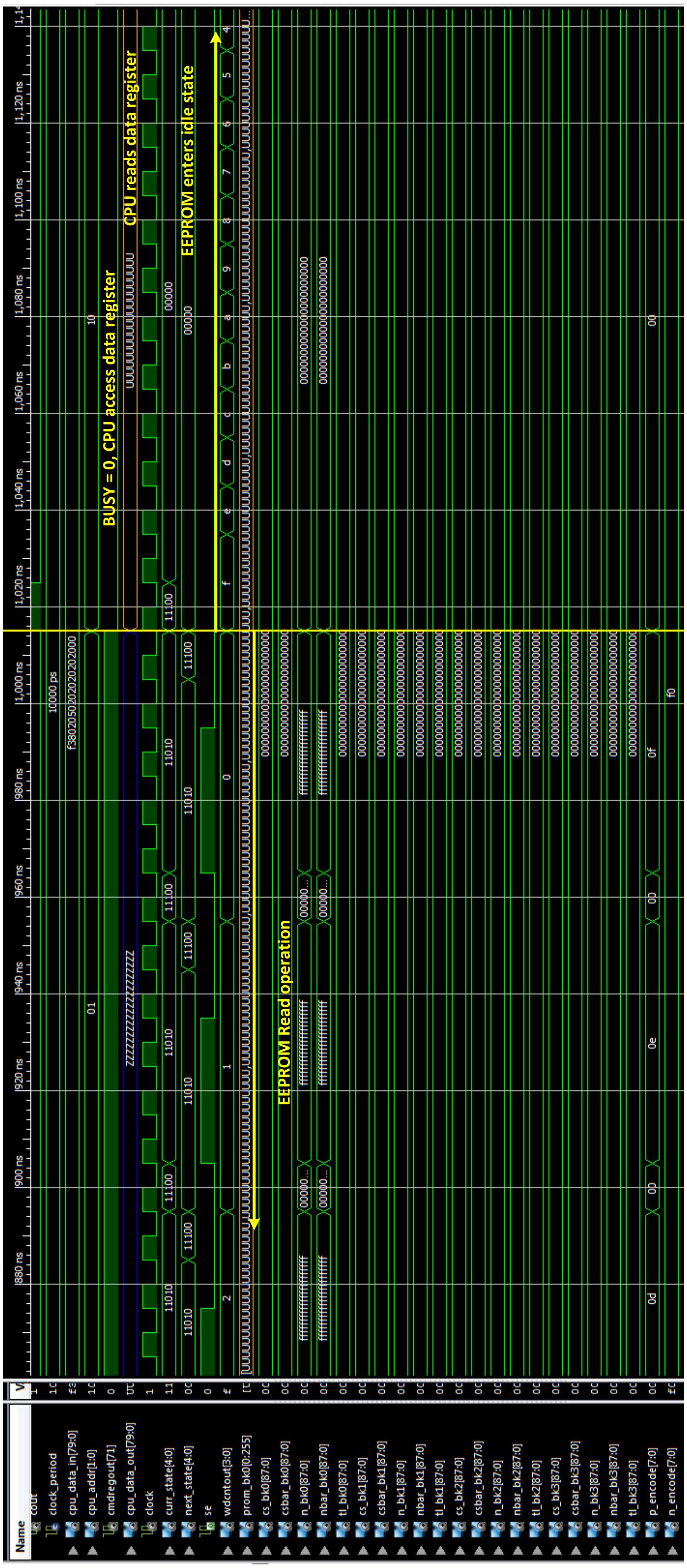
Figure 5.3 Waveforms of the read operation.



Read a block (16 words)

This waveform shows the transition from CPU access of the data register to EEPROM read.

SE, sense amplifier enable signal is high for each read operation.



BUSY = 0, CPU access data register

CPU reads data register

EEPROM enters idle state

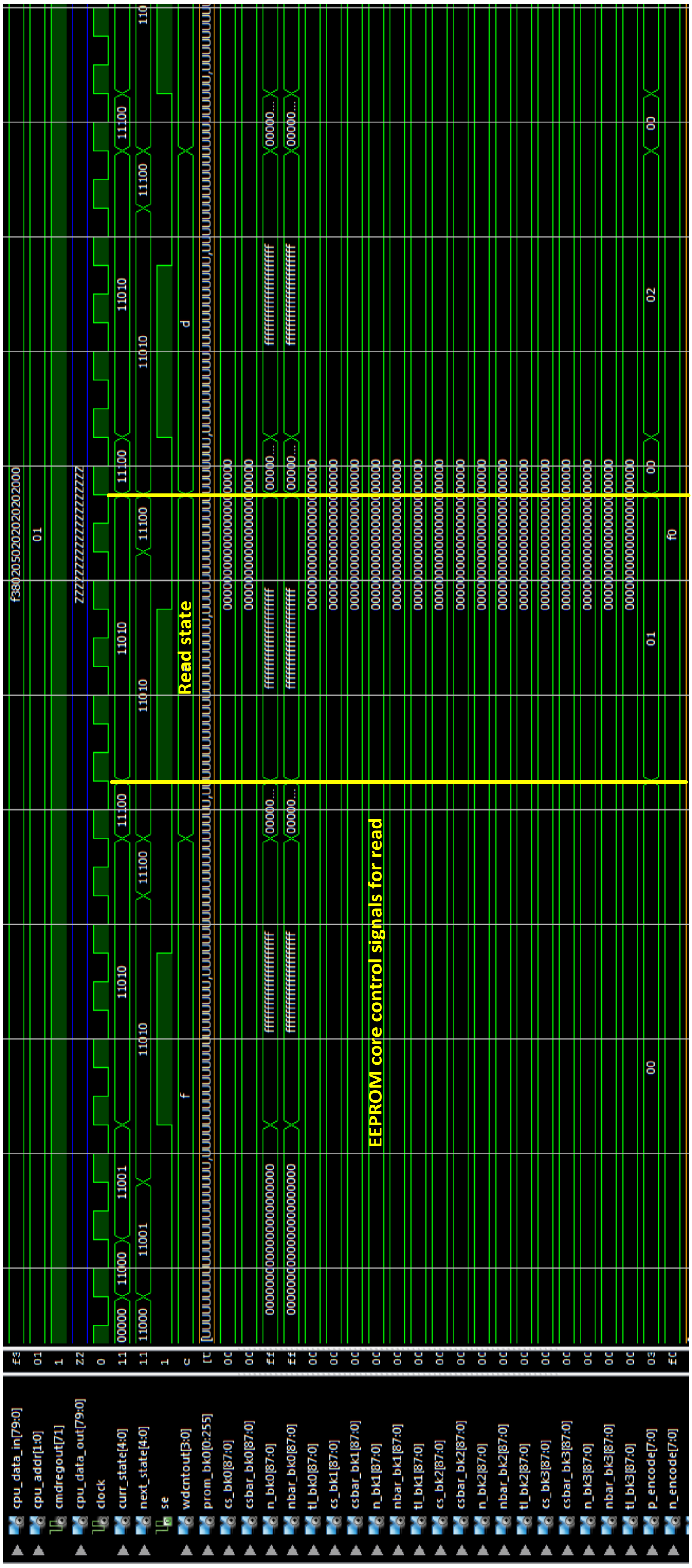
EEPROM Read operation

Read a block (16 words)

This waveform shows the transition from EEPROM read operation to CPU reobtaining data register access.

After EEPROM read is complete, BUSY bit is cleared showing that the CPU has access to the data register and EEPROM control enters idle state.

After the CPU regaining access to the data register, it reads out the content of the data register for CPU retrieval and confirmation.



This waveform shows the EEPROM core control signal status for read operation.

Read bank

CS_Bk# = 0

CSbar_Bk# = 0

N_Bk# = 1

Nbar_Bk# = 1

TL_Bk# = 0

Non Read bank

CS_Bk# = 0

CSbar_Bk# = 0

N_Bk# = 0

Nbar_Bk# = 0

TL_Bk# = 0

5.2 Wrapper Synthesis Report

Reports are generated through the synth.out code file which is then run through the DC compiler on the server using TSMC 180nm cell library technology. The input for the synthesis code is the top-level wrapper vhdl file. The synthesis output gives the reports on power requirements, timing enclosure, total area and number of gates and components. The synthesis also generates a netlist (.vh file) which can be used for importing the interface design into encounter tool to perform place and route. The report includes 3 sub-reports namely:

- Total number of gates and components used for the wrapper design, namely macro statistics.
- Power report that includes all the internal power, switching power and leakage power stated clearly. Also, all the powers are defined in table with respective to the power group like sequential, combinational, io pads, memory, register and black-box.
- Area report that documenting total area required for the design which is the summation of combinational area, non-combinational area, buf/inv area, black-box area and net interconnect area.

5.2.1 Gates and Components Report

HDL Synthesis Summary is as follows:

```
inferred 90816 D-type flip-flop(s).
inferred 32 Comparator(s).
inferred 352 Multiplexer(s).
=================================================================
```

HDL Synthesis Report:

Table 5.1 Macro Statistics of gate count.

# Counters	9
32-bit down counter	7
8-bit down counter	1
8-bit up counter	1
# Registers	1058
1-bit register	6
5-bit register	1
71-bit register	1
8-bit register	1
80-bit register	17
88-bit register	1032
# Latches	10
1-bit latch	2
3-bit latch	4
80-bit latch	4
# Comparators	32
88-bit comparator equal	24
88-bit comparator not equal	8
# Multiplexers	7
80-bit 16-to-1 multiplexer	1
80-bit 4-to-1 multiplexer	2
88-bit 256-to-1 multiplexer	4
# Decoders	3
1-of-16 decoder	1
1-of-4 decoder	2
# Tractates	3
80-bit tristate buffer	3
# Xors	2
1-bit xor2	2

Final Macro Processing:

==

Final Register Report:

Macro Statistics:

# Registers	83787
Flip-Flops	83787

==

```
Device utilization summary:

Number of Slices:                72297 out of   4656   1552%
Number of Slice Flip Flops:      84041 out of   9312   902%
Number of 4 input LUTs:          132781 out of   9312   1425%
Number of IOs:                   169
Number of bonded IOBs:           169 out of    232    72%
Number of GCLKs:                 4 out of     24    16%
```

5.2.2 Power Report

```
Design: toptest
Version: M-2016.12-SP1
Date: Tue Jul 10 11:41:51 2018
Library(s) Used:
osu018_stdcells (File: /home/chenha/placeroute/hutchens/lib/tsmc018/lib/osu018_stdcells.db)
Operating Conditions: typical   Library: osu018_stdcells
Wire Load Model Mode: top
Global Operating Voltage = 1.8
Power-specific unit information:
Voltage Units = 1V
Capacitance Units = 1.000000pf
Time Units = 1ns
Dynamic Power Units = 1mW (derived from V, C, T units)
Leakage Power Units = 1nW
Cell Internal Power       = 269.216003nW (77%)
Net Switching Power    = 81.774673nW (23%)
```

Total Dynamic Power   = 350.990692nW (100%)

Cell Leakage Power   = 409.0789pW

Information: report_power power group summary does not include estimated clock tree power. (PWR-789)

Table 5.2 Power report of the interface architecture.

Power Group	Internal Power (mW)	Switching Power (mW)	Leakage Power (nW)	Total Power(mW)	(%)	Attrs
Io_pad	0	0	0	0	0	
Memory	0	0	0	0	0	
Black_box	0	0	0	0	0	
Clock_network	0	0	0	0	0	
Register	0	0	0	0	0	
Sequential	1.985734e-04	1.618086e-06	0.277270	2.004688e-04	57.05	
Combinational	7.064256e-05	8.015659e-05	0.131809	1.509310e-04	42.95	
Total	2.692160e-04	8.177468e-05	0.409079	3.513998e-04		

5.2.3 Area Report

Design: toptest

Version: M-2016.12-SP1

Date: Tue Jul 10 11:41:51 2018

Library(s) Used:

osu018_stdcells (File: /home/chenha/placeroute/hutchens/lib/tsmc018/lib/osu018_stdcells.db)

Number of ports:	5
Number of nets:	10
Number of cells:	6
Number of combinational cells:	4

```
Number of sequential cells:           2

Number of macros/black boxes:         0

Number of buf/inv:                    2

Number of references:                 4

Combinational area:                   79.000000

Buf/Inv area:                         32.000000

Non-combinational area:               176.000000

Macro/Black Box area:                 0.000000

Net Interconnect area:                undefined (No wire load specified)

Total cell area:                      255.000000

Total area:                           undefined

The above information was reported from the logical library. The following are from the physical
library:

Hierarchical area distribution

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

                          Global cell area                   Local cell area

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

Hierarchical cell        Absolute   Percent      Combi-        Noncombi-      Black-

                            Total      Total         national      national        boxes
Design

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

toptest                  255.0000   100.0        79.0000         176.0000         0.0000
toptest

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

Total                                                    79.0000         176.0000         0.0000

Total synthetic cell area:   0.0000       0.0%
```

```
Cell report

Design: toptest

Version: M-2016.12-SP1

Date: Tue Jul 10 11:41:51 2018

****************************************

Attributes:

    b - black box (unknown)

    h - hierarchical

    n – non-combinational

    r - removable

    u - contains unmapped logic


Cell                Reference           Library                   Area

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

Q_reg              DFFSR              osu018_stdcells           176.00000

U7                  INVX1              osu018_stdcells           16.000000

U9                  INVX1              osu018_stdcells           16.000000

U10                 NAND2X1         osu018_stdcells           24.000000

U11                 OAI21X1         osu018_stdcells           23.000000

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

Total 5 cells                                                   255.000000
```

5.2.4 Timing Report

```
Operating Conditions: typical   Library: osu018_stdcells
Wire Load Model Mode: top

  Startpoint: CPU_addr[1] (input port clocked by vclk)
  Endpoint: CPU_data_out[14] (output port clocked by vclk)
  Path Group: vclk
  Path Type: max

  Point                          Fanout    Cap       Trans      Incr       Path
  -----------------------------------------------------------------------------------
  clock vclk (rise edge)                                        0.000000   0.000000
  clock network delay (ideal)                                   0.000000   0.000000
  input external delay                                          0.000000   0.000000 r
  CPU_addr[1] (in)                                   0.097514   0.064888   0.064888 r
  CPU_addr[1] (net)                5       0.059322             0.000000   0.064888 r
  U9396/Y (INVX1)                                    0.120321   0.120657   0.185545 f
  n10245 (net)                     4       0.055949             0.000000   0.185545 f
  U16378/Y (INVX1)                                   0.685997   0.537376   0.722921 r
  n10561 (net)                     17      0.287039             0.000000   0.722921 r
  U16379/Y (NOR2X1)                                  0.921447   1.090908   1.813828 f
  n10637 (net)                     30      0.492404             0.000000   1.813828 f
  U16380/Y (BUFX2)                                   0.793983   0.875437   2.689266 f
  n10655 (net)                     52      0.792021             0.000000   2.689266 f
  U16748/Y (NAND2X1)                                 0.146019   0.231521   2.920787 r
```

```
n10607 (net)                  1    0.012903               0.000000   2.920787 r

U16749/Y (OAI21X1)                           0.193626    0.061718   2.982504 f

n10609 (net)                  1    0.015080               0.000000   2.982504 f

U16750/Y (AOI21X1)                           0.154865    0.096712   3.079216 r

n349 (net)                    1    0.017097               0.000000   3.079216 r

output_tri[14]/Y (TBUFX1)                    0.087053    0.044609   3.123825 f

CPU_data_out[14] (net)        1    0.004537               0.000000   3.123825 f

CPU_data_out[14] (out)                       0.087053    0.000000   3.123825 f

data arrival time                                                   3.123825


clock vclk (rise edge)                                   100.0000   100.000000

clock network delay (ideal)                              0.000000   100.000000

output external delay                                    0.000000   100.000000

data required time                                                  100.000000

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

data required time                                                  100.000000

data arrival time                                                   -3.123825

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

Slack (MET)                                                       96.876175
```

On the macro level, synthesis reports summarize the architecture having a gate count of 90816 D-type flip-flops, 32 comparators and 352 multiplexers. Total dynamic power consumption is less than 400nW with total synthetic cell area of 255 square microns based on TSMC 180nm geometry. If scaled down to smaller technology process such as 40 nm node, power consumption is reduced to ~35nW with cell area of ~20 square microns.

CHAPTER VI

CONCLUSION

A custom interface for Charge Trap EEPROM application is proposed. The architecture includes full data communication interface between CPU and EEPROM core. The importance of the proposed structure is that the implementation of the EEPROM is on-chip where on-chip processing minimizes the vulnerability from boot attacks. Since the data is moved from off-chip to on-chip and its transport is in parallel further preventing the success of side channel attacks using power analysis.

Regarding the future trend of HfO_2 scalability as a dielectric trapping layer, it is observed that Hf-based dielectrics have higher breakdown than SiO_2 and properties can be enhanced by incorporating other metals such as Al. The rare earth oxides, various lanthanides, their silicates and recently rare earth scandates can also be counted to potentially replace HfO_2 for MOS dielectric [14] [15]. According to International Technology Roadmap for Semiconductor (ITRS), reduction of the EOT will continue to be a difficult challenge in the near term with respect to High-k Metal Gate as well [4] [13]. So new device architecture such as multiple-gate MOSFETs (e.g., FinFETs) and Gate-All-Around (GAA) are expected to allow scaling beyond the 10nm node [4] [14] [15].

Based on available data from the literatures [1] [6], a programming range for write operation is 1.5V to 2.7V and erase gate voltage is 0 to -1.5V resulting -1.5V to -3V across gate source. These nominal programming voltages are optimal in terms of data retention and lifetime. Therefore, we propose a range of programming voltages that could give optimal results.

The logic correctness of the EEPROM wrapper is checked against a VHDL EEPROM memory model that we developed based on charge-trap memory cell operations. The model consists of four 256 by 88 arrays. Each array represents one EEPROM bank. Write/Read/Erase operation of the array is performed based on the control signals generated in the wrapper. A Xilinx simulator platform is used to validate the VHDL code. The CUI interface architecture area is negligible compared to the memory size.

The fully implemented on-chip CPU-EEPROM interface enables secure EEPROM core programming control and data communication with the CPU. Therefore, it is possible to achieve secure communication between devices without interface eavesdropping. Secure encryption keys and full-data security are attained.

REFERENCES

[1] C. Kothandaraman et al, “Oxygen vacancy traps in Hi-K/Metal gate technologies and their potential for embedded memory applications,” 2015 IEEE Internatioinal Reliability Physics Symposium, Monterey, CA, pp. MY.2.1-MY.2.4., 2015.

[2] J. Halderman et al., “Lest we remember: cold-boot attacks on encryption keys,” Communications of the ACM, vol. 52(5), pp. 91–98, May 2009.

[3] Owen Lo et al., “Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power naalysis (CPA),” Journal of Cyber Security Technology 2017, 1:2, 88-107.

[4] Saeed Mohsenifar, M. H. Shahrokhabadi., "Gate Stack High-κ Materials for Si-Based MOSFETs Past, Present, and Futures," in Microelectronics and Solid State Electronics 2015, 4(1): 12-24.

[5] 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,” IEEE Electron Device Letters, vol. 38, no. 1, pp. 44-47, Jan. 2017.

[6] B. Jayaraman et al., “80-kb logic embedded high-k charge trap transistor-based multi-time-programmable memory with no added process compliexity,” IEEE Journal of Solid-State Circuits, vol. 53, no. 3, pp. 949-960, March 2018.

[7] Chun Zhao *et al*., "Review on Non-Volatile memory with High-k dielectrics: Flash for generation beyond 32nm,” in High-k materials and devices, Materials 2014, MDPI.

[8] Sergei skorobogatov, “Local heating attacks on flash memory devices,” 2009 IEEE International Workshop on Hardware-Oriented Security and Trust (HOST).

[9] Hirotaka Hamamura et al., “Electron Trapping Characteristics and Scalability of HfO_2 as a Trapping Layer in SONOS-type Flash Memories,” IEEE 46th Annual International Reliability Physics Symposium, Phoenix, 2008.

[10] Janakiraman Viraraghavan et al., “80Kb 10ns Read Cycle Logic Embedded High-K Charge Trap Multi-Time-Programmable Memory Scalable to 14nm FIN with no Added Process Complexity,” 2016 IEEE Symposium on VLSI Circuits Digest of Technical Papers.

[11] T. Sugizaki et al., “Novel Multi-bit SONOS Type Flash Memory Using a Highk Charge Trapping Layer,” 2003 IEEE Symposium on VLSl Technology Digest of Technical Papers.

[12] Yee-Chia Yeo et al., “Direct tunneling leakage current and scalability of alternative gate dielectrics,” Appl. Phys. Lett. 81, 2091 (2002).

[13] E.P. Gusev et al., “Charge detrapping in HfO2 high-k gate dielectric stacks,” Appl. Phys. Lett. 83, 5223 (2003).

[14] Kelin J. Kuhn et al., “The Ultimate CMOS Device and Beyond,” 2012 IEEE International Electron Devices Meeting.

[15] Subramanian S. Iyer, “The Evolution of Dense Embedded Memory in High Performance Logic Technologies,” 2012 IEEE International Electron Devices Meeting.

[16] C. Kothandaraman et al., “Electrically Programmable Fuse (eFUSE) Using Electromigration in Silicides,” IEEE Electron Device Letters, vol. 23, no. 9, September 2002.

VITA

Vishal Reddy Banala

Candidate for the Degree of

Master of Science

Thesis: CUSTOM INTERFACE FOR CHARGE TRAP EEPROM APPLICATIONS

Major Field: Electrical and Computer Engineering

Biographical:

Education:

Completed Master of Science degree in Electrical and Computer Engineering at Oklahoma State University, Stillwater, Oklahoma in May, 2019.

Completed Bachelor of Technology degree in Electronics and Communication Engineering at Jawaharlal Nehru Technological University, Hyderabad, Telangana, India in April, 2017.

Experience:

Research Assistant at Oklahoma State University, Jan. 2018 – May 2019.