Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ce.et.tudelft.nl/

2012

MSc THESIS

Flash Memory Device: Electrical Modeling and Simulation

Ronnie Klanderman

CE-MS-2012-15

Abstract

Flash memory, created in the early eighties and based upon EEPROM, has become a very popular non-volatile memory since the start of the millennium. A Flash memory cell consists of 1 MOS transistor with a floating gate. The floating gate (in between the channel and the control gate) is able to trap electrons due to writing mechanisms (e.g. Fowler Nordheim and Channel Hot Electron Injection) and hereby changing the threshold level of the transistor. Flash also has different cell array architectures that define its read and write speed, and geometrical size. The array architectures called NOR and NAND are currently most popular. NOR is advantageous for high speed read and writing, however the NAND architecture is more compact and therefore more suitable for mass data storage. By creating a functional model of a Flash memory device, design of an electrical Flash memory device and its simulation can be simplified. After studying Flash cell models in literature, a Flash cell model and complete memory device will be presented in this Thesis. This study has also led to distinction between "Static" and "Dynamic" Flash cell models. Static models need a voltage or current source to change the threshold voltage. This results in 2 models, one for writing and another for reading. Dynamic models can use equations as behavioral blocks to ensure a threshold voltage change at the transistor and have one model to represent both reading and writing. In order to show simulation results of a Flash memory device, a 2x2 bit NOR and 1x2 bit NAND will be presented for analysis. Both with a dynamic cell model. Fowler Nordheim and Channel Hot Electron Injection are also taken into account in the presented net lists. The simulations have been done in HSPICE, an industrial version of SPICE (Simulation Program with Integrated Circuit Emphasis).

Faculty of Electrical Engineering, Mathematics and Computer Science

Flash Memory Device: Electrical Modeling and Simulation

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Ronnie Klanderman
born in Mombasa, Kenia

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Flash Memory Device: Electrical Modeling and Simulation

by Ronnie Klanderman

Abstract

Flash memory, created in the early eighties and based upon EEPROM, has become a very popular non-volatile memory since the start of the millennium. A Flash memory cell consists of 1 MOS transistor with a floating gate. The floating gate (in between the channel and the control gate) is able to trap electrons due to writing mechanisms (e.g. Fowler Nordheim and Channel Hot Electron Injection) and hereby changing the threshold level of the transistor. Flash also has different cell array architectures that define its read and write speed, and geometrical size. The array architectures called NOR and NAND are currently most popular. NOR is advantageous for high speed read and writing, however the NAND architecture is more compact and therefore more suitable for mass data storage. By creating a functional model of a Flash memory device, design of an electrical Flash memory device and its simulation can be simplified. After studying Flash cell models in literature, a Flash cell model and complete memory device will be presented in this Thesis. This study has also led to distinction between "Static" and "Dynamic" Flash cell models. Static models need a voltage or current source to change the threshold voltage. This results in 2 models, one for writing and another for reading. Dynamic models can use equations as behavioral blocks to ensure a threshold voltage change at the transistor and have one model to represent both reading and writing. In order to show simulation results of a Flash memory device, a 2x2 bit NOR and 1x2 bit NAND will be presented for analysis. Both with a dynamic cell model. Fowler Nordheim and Channel Hot Electron Injection are also taken into account in the presented net lists. The simulations have been done in HSPICE, an industrial version of SPICE (Simulation Program with Integrated Circuit Emphasis).

Laboratory	:	Computer Engineering
Codenumber	:	CE-MS-2012-15

Committee Members :

Advisor:	Dr. Ir. Zaid Al-Ars, CE, TU Delft
Chairperson:	Dr. Koen Bertels, CE, TU Delft
Member:	Dr. Ryoichi Ishihara, EE, TU Delft
Member:	Dr. Ir. Michiel Pertijs, EE, TU Delft

Contents

List of Figures

List of Tables

Acknowledgements

Three months before I was supposed to graduate in 2009 I took on a job as Software Engineer with a contracting company. It was just before the financial crisis hit our economy. I was happy to get a good offer because it somewhat changed after this event and good offers in the IT market declined for people fresh from University or College.

This choice resulted in me not finishing my Thesis that year and due to working on my career and experiencing all the changes in life, I procrastinated and almost quit. I still had to finish 2 more chapters to complete it. 2010 came and went and I still didn't manage to do anything. I think being ashamed for being idle and very busy with work at the same time prevented me from discussing it with my supervisor. Prolonging it felt safe until it became too painful. Somehow I finally contacted my supervisor and to my surprise he gave me another chance to complete my work. "It's up to you" he said. He was right. Making a choice can change everything.

Thank you Dr. Ir. Zaid Al-Ars for giving me this second chance and for our many late night discussions. Memories of going to class have been replaced by packing dinner and eating it while driving to Delft after my work day ended. We would then discuss models and I learned a lot from how you structure and reason. Also, I would like to thank Marjolijn for her patience and support. Last but not least, thank you Mother for all you have done for me. Thanks everyone.

Ronnie Klanderman
Delft, The Netherlands
October 8, 2012

Introduction 1

We live in a world of growing possibilities regarding communication and computation. All this is made possible by computers with three general features: data exchange (in/out), data processing (logic) and data storage (memory). As the field of Computer Engineering keeps on advancing, computers are becoming more efficient at all three of these features. The future of computing devices is one where memory (which is already a substantial part of many devices) will increase even more with respect to logic. [6] While facilitating more memory on integrated circuits, we also aim at improving their general features. When storing more data however we also tend to create more faults in the creation of memory cells due to complicated processes. To study these kind of introduced and other faults, we can use electrical models to simulate them.

Flash memory devices were introduced in the eighties [3] and have become quite popular since then. Increasingly more products are using Flash memory technology. It is becoming more difficult to name digital devices without Flash memory. Flash's popularity is mainly due to its property of retaining binary digits (*bits*) without the need of a continuous power source. As soon as it is written (programmed or erased), it only requires power for its bits to be read. In this Thesis, we discuss the development of an electrical Flash memory model.

In the first section of this chapter, we will look at general memory technologies. In Section 1.2 we will explore transistor based memory devices and in Section 1.3 we will look at Flash memory technology. In Section 1.4 we will look at the objectives of this project and we will end this chapter with a section that outlines the rest of the Thesis.

1.1 Memory Technology

Memory for computing devices can be categorized in several hierarchies. We categorize memory type to specify the different storage locations of data during the processing of an instruction. In Figure 1.1 we distinguish four levels in this hierarchy of types. Register level is shown at the top and is considered to be the fastest accessible memory. At the lower end of Figure 1.1 we see the slowest accessible memory.

We can classify memories in three main memory technology categories shown in Figure 1.2. We will briefly discuss these memory technologies and go into more details of transistor based memory devices in Section 1.2. Flash memory is transistor based.

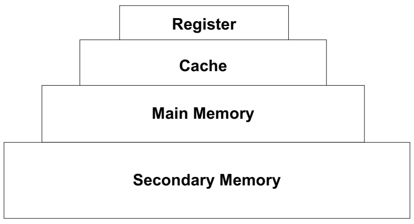


Figure 1.1: memory hierarchy levels

1.1.1 Magnetic Devices

Magnetic memory devices hold the highest market share in the computer industry and is also the oldest data storage technology. Magnetic tape devices store large quantities of data at relatively low costs. It uses an inductive head that translates incoming electrical signals to a magnetic field which is captured on moving tape. This tape is sensitive to the magnetic variations, thus registering data. However, the performance of magnetic tape is far from ideal and can wear out because of the constant exposure to the inductive head. By innovating magnetic tape, the idea of a magnetic disk came to pass. It uses a low mass slider. This slider has an inductive head that floats at a precise distance from the magnetic medium. Magnetic disks are still the most common storage devices in desktop computers. However, solid state disks (a Flash memory device) are gaining in popularity and will soon probably replace magnetic disks in computer sales. Magnetic memory devices generally represent the lower part of the memory hierarchy system depicted in Figure 1.1. We call this secondary memory.

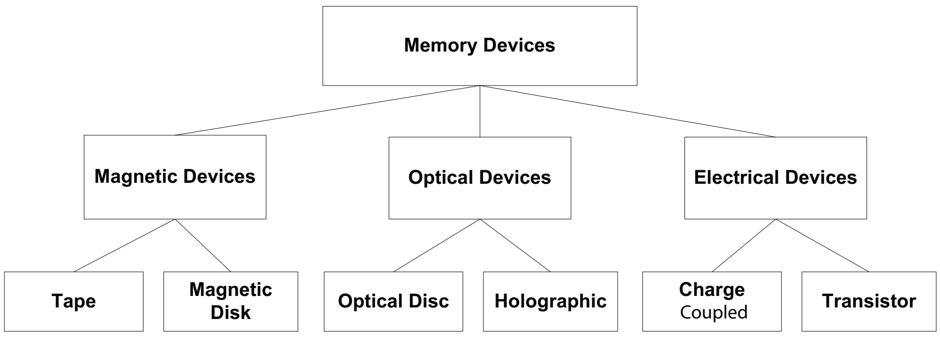


Figure 1.2: main memory technology categories

1.1.2 Optical Devices

Optical memory devices are also well represented as storage devices in the computer industry. Most popular are the optical disc's (e.g. CD-ROM, DVD-ROM). When buying software, e.g. office applications and games, consumers mostly acquire it via the optical disc. With the disc, data is written onto the surface by using dots and spaces. A laser beam then detects the stream of bits by their reflection while rotating. As for holographic storage devices, they mainly store data as images in crystals by using a laser beam to change the refractive index of the crystal. Similar to magnetic memory devices, optical memory devices are part of the lowest memory hierarchy shown in Figure 1.1.

1.1.3 Electrical Devices

Of the electrical devices category, the *CCD* (Charge Coupled Devices) is quite a niche in the computer industry. CCD's control the electrical charge from and to a memory element. The electrical charge in this device determines the logic level it holds. CCD's are also serially accessed, which is slower than random access of memory cells in parallel. We commonly find these devices in the field of digital imaging and they are also placed in the lower memory hierarchy shown in Figure 1.1.

For fast read and write times regarding data storage, transistor based devices are better suited than the previous ones. In fact, they have the best read and write time compared to all the other devices discussed. Transistor based memory devices are used extensively on and near data processing units. These devices dominate the upper three parts of the memory hierarchy in Figure 1.1. We will look at the transistor based memory devices in the next section. Transistor based memory is made in the semiconductor industry. In the semiconductor industry, memory devices are either categorized as *RAM* (Random Access Memory) or *ROM* (Read Only Memory) as shown in Figure 1.3. Also, semiconductor memory devices can be distinguished in two types, namely *volatile* and *non-volatile* memory. RAM memory has a volatile nature and ROM memory non-volatile. Volatile memory need continuous power to retain data, i.e. the transistor based circuit requires power to keep the data of its cells. Non-volatile memory retains data after it has been written (i.e. programmed or erased). A cell represents the smallest quantity of data in a memory device, i.e. a bit. Flash memory cells however are capable of storing n bits in a single cell. We will discuss this concept later after we discuss the different memory types depicted in Figure 1.3 in the next section.

1.2 Transistor Based Memory Devices

Transistors are elementary building blocks in computer systems. They function as switches and by grouping specific types we obtain the logic and memory that builds an electrical system. The transistors used in high density memory devices such as Flash memory devices are called *MOS* (Metal Oxide Semiconductor) transistors. MOS transistors are available in two types, *NMOS* (Negative MOS) where a given voltage at the gate causes it to conduct electricity between gate and source, and a voltage below this given value closes it from conducting. *PMOS* (Positive MOS) works the other way around, where a given voltage at the gate stops it form conducting electricity between gate and source. NMOS is currently more used due to its operational speed. It switches faster. Both MOS transistors combined make a *CMOS* (Complementary MOS) circuit and this combination is frequently used in the current semiconductor

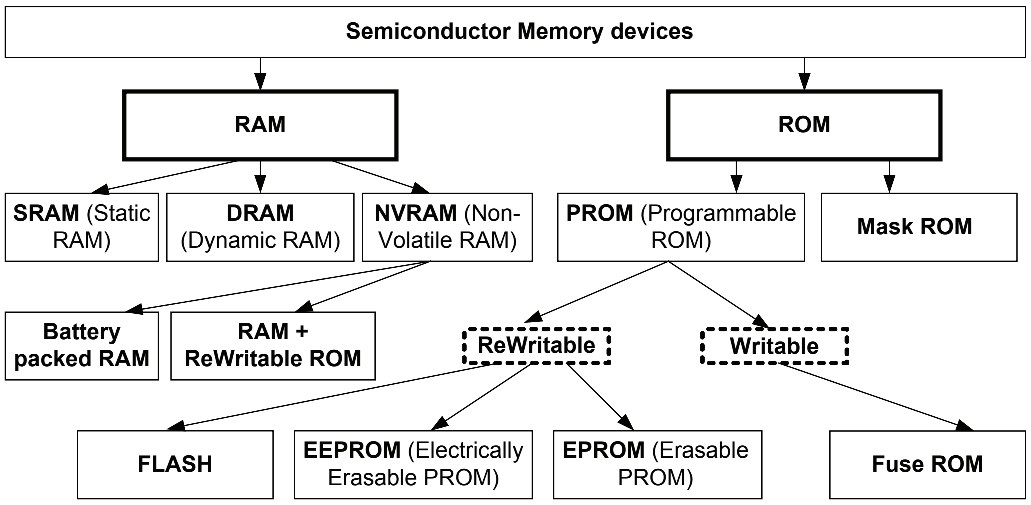


Figure 1.3: RAM and ROM devices

industry. The basic layout of a CMOS circuit is depicted in Figure 1.4. CMOS circuits are especially popular to use due to low power consumption. Figure 1.4 is either pulled up or down. By constructing combinations of CMOS circuits we are able to create logical gates, e.g. NAND and NOR. In turn, these logical gates are the building blocks of contemporary digital circuits that make up our computing systems.

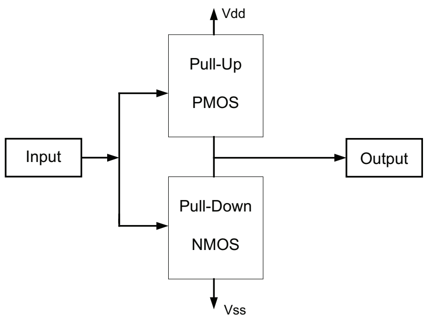


Figure 1.4: CMOS circuit

Let us look at the *write ability* of memory. This is the number of times that a memory can be written, e.g. programmed (i.e. logical 1) or erased (i.e. logical 0). We can also look at memory in terms of *storage permanence*, which refers to the duration of reliably retaining data in memory cells when power is disconnected. Intuitively, we can say that an ideal memory device is one with an unlimited write ability and infinite storage permanence. Figure 1.5 depicts an overview of write ability and storage permanence of transistor based memories.

Figure 1.5 shows that *SRAM* (Static RAM) and *DRAM* (Dynamic RAM) have an unlimited write ability. However the storage permanence is almost zero in time, thus

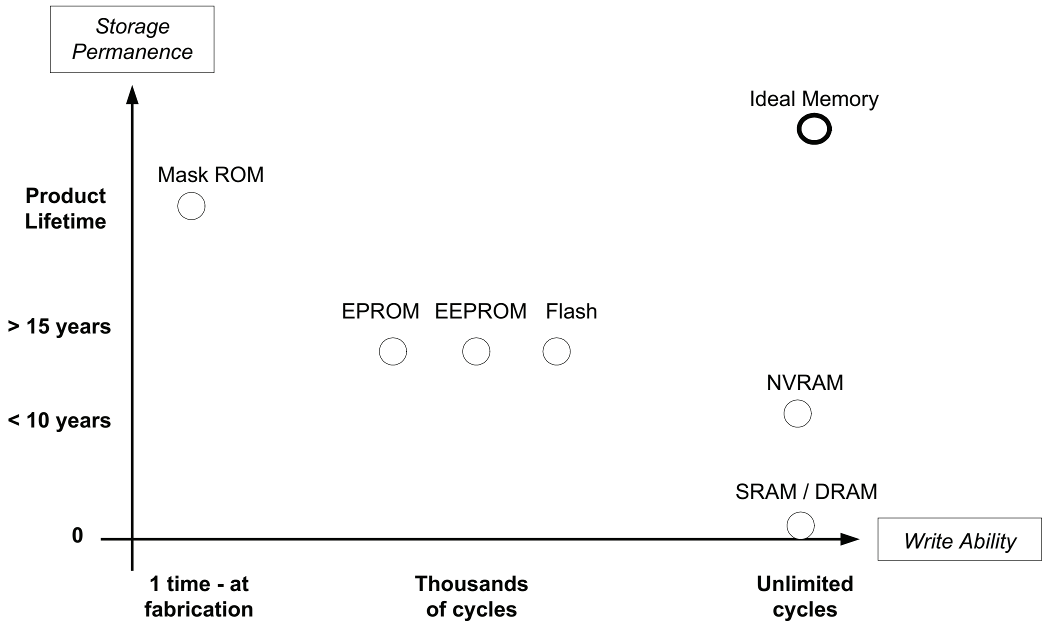


Figure 1.5: storage permanence and write ability of transistor based memory; [21]

marking these memories as volatile devices. Flash is limited to thousands of cycles of data writing. Its storage permanence results in at least a decade of reliable data storage, marking Flash as a non-volatile memory device. We will briefly discuss the memories depicted in Figure 1.5 and go into more details regarding Flash memory in the next section.

Mask ROM is by far the most durable memory type with the lowest programmability. The internal logic is programmed at fabrication time by using a set of masks. The memory lookup table that will be hard wired is one that must be tested extensively before fabrication and the number of these memory chips produced for each programmed lookup table should be relatively large because of the production costs.

EPROM (Electrically Programmable ROM) is a programmable ROM with a transistor that uses an *FG* (Floating Gate). An FG is able to trap charge on an isolated plate. This plate is surrounded by an oxide layer that traps electrons. The electrons are injected into the FG by using a high programming voltage, approximately 20 V. To erase data, this charge must be drained from the FG. An EPROM has an erase window that enables charge drainage by exposing the cell to rays of ultraviolet light for a certain time, i.e. 5 to 30 minutes. An EPROM is written in total. This means all memory cells are involved in the write process. Table 1.1 depicts the characteristic of this transistor based memory.

An *EEPROM* (Electrically Erasable PROM) is programmed and erased electrically. The efforts put into making EPROM electrically erasable has led to the EEPROM. Like the EPROM, it also has an FG that can trap and drain charge. EEPROM devices can write words of data, which is more effective than the EPROM. Also, due to its electrical erasing and programming of words, write times are considerably faster. Table 1.1 depicts the characteristic of this memory type.

Flash memory is an extension of the (x)PROM family. Similar to the others it uses an FG to trap charge. Flash devices can program or erase individual blocks of words. Making it even more effective than the EEPROM. This capability can improve write times when dealing with a large number of data. Table 1.1 depicts the characteristic of this transistor based memory. We will discuss more of the Flash cell in Chapter 2.

SRAM uses a flip-flop structure of generally 4 to 6 transistors to make up a cell. The name static is coined because this memory stores data (logical 0 or 1) as long as the transistors are powered. SRAM is used for high performance data traffic, e.g. cache memory, due to its fast write times. SRAM is represented in the upper 2 memory hierarchy levels depicted in Figure 1.1. Its characteristics are also depicted in Table 1.1.

DRAM uses a transistor and a capacitor to store cell data. This charge storage however is of temporary nature and the capacitor needs to be recharged due to the gradual leakage of power. Its design is very compact compared to SRAM but needs refreshing. A DRAM cell's refresh rate is in the magnitude of microseconds. The third memory hierarchy level namely main memory, is represented by DRAM. This is shown in Figure 1.1. Table 1.1 also depicts the characteristic of this transistor based memory.

NVRAM (Non-Volatile RAM) devices can be divided in two groups, the battery packed RAM and the combination of RAM with ROM. The battery packed RAM contains an SRAM with its own power supply that can last for as long as 10 years. The RAM with ROM combination stores its complete RAM content into an EEPROM just before the power supply is turned off. It reloads this content from the EEPROM back into the RAM when the power goes back on. These hybrid memory types may prove to be advantageous in the near future.

Table 1.1: characteristics of different transistor based memories; [1] [4] [18] [21]

Criterion	EPROM	EEPROM	Flash	SRAM	DRAM
Cell size	1-2	3-4	1	4-6	1-2
Memory type	non-volatile	non-volatile	non-volatile	volatile	volatile
Write ability	$n\ 10^4$	$n\ 10^4$	$n\ 10^5$	∞	∞
Storage permanence	10+ yrs	10+ yrs	10+ yrs	n ns	n ms
In-system writing	no	yes	yes	yes	yes
Typical write speed	-	2-3 s	1 s	15 ns	60 ns
Typical read speed	90 ns	200 ns	100 ns	15 ns	60 ns

1.3 Introduction to the Flash Cell

The principles memory cells with an FG were successfully put into practice in the late sixties and resulted in the (x)PROM family. In the late seventies, Toshiba was working on a new version of EEPROM. [8] At the time, the EEPROM was an expensive memory type and a major challenge was to reduce its production costs. This was achieved by reducing the number of transistors that make up the cell and has led to the *1T* (1 Transistor) Flash cell. The 1T Flash cell was complemented with a multi-byte writing scheme and was given the name 'Flash' because it was possible to erase a complete array of cells.

Figure 1.6 depicts 1T cell storage device. The main difference between a MOS transistor and 1T Flash is the capacitive FG. The gate of a 1T Flash, *CG* (Control Gate), is used for writing the cell by influencing the charge on the FG. Similar to an NMOS transistor, the industry standard 1T Flash has an N-type source and drain that are doped with donor atoms. The P-type substrate is doped with acceptor atoms. The FG and CG are doped with donor atoms and are divided by insulating oxide layers. Like a MOS transistor, Flash cells can be put in *cutoff* (CO), *triode* (TR) and *saturation* (SR) mode. [17] The general way of programming a Flash cell is by setting CG to a high voltage. This results in a pull of electrons from the substrate of the 1T Flash. As for erasing, the CG is set to low and the source of the 1T Flash is set to a high voltage. This results in a drainage of electrons toward the source. Flash cells are a remarkable product of CMOS process technology. We will have a more detailed look at the 1T Flash cell in the next chapter.

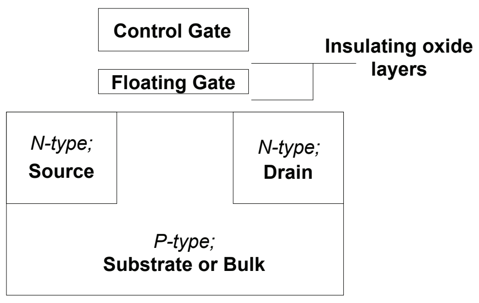


Figure 1.6: basic 1T cell storage device

A simple scheme for programming and erasing a Flash cell is depicted in Figure 1.7. For a Flash memory cell to be programmed, source is set to GND and Control Gate is set to a high voltage, Vdd. We see that electrons are pulled up toward the FG by getting enough energy to be pulled through the oxide layer. They become trapped in the FG. For erasing, we see that the trapped charge (electrons within the FG) is pulled toward the source through the oxide layer. The charge is then removed from FG, rendering the cell erased. The charged FG, which represents a programmed cell, has a higher threshold voltage than an erased cell. This is due to the trapped electrons on the FG that require the CG voltage to be higher in order to set the cell in saturation mode. The capacitive nature of the FG gives this functionality by trapping or draining charge.

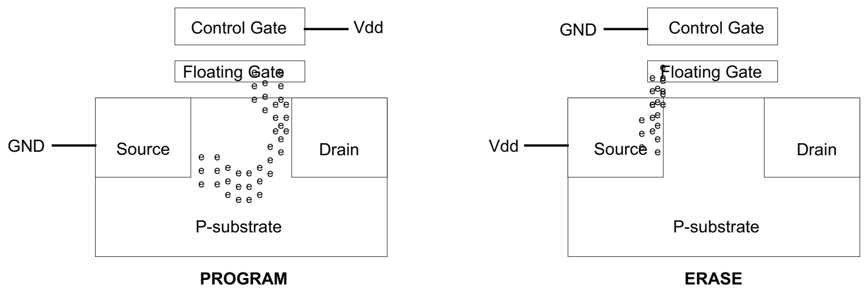


Figure 1.7: writing a 1T flash cell storage device

Flash memory devices have different cell array architectures. Popular Flash architectures are NOR and NAND. The adapted Flash architecture determines how a cell is read and written. These architectures result in different memory characteristics, e.g. read and write times. Flash memory manufacturers choose an architecture based upon its characteristic. NOR and NAND are the most used Flash array architectures. NOR is characterized by its fast data transfers, but it is slower to program and erase than the NAND. It also has a larger cell area than a NAND.

1.4 Thesis Objective and Outline

This Thesis has been written in order to create an electrical Flash memory model. The objective is to develop an electrical model of a Flash memory device, based upon a functional model and the industry standard Flash memory cell. Chapter 2 targets the Flash cell and cell array architectures. Chapter 3 will further explore the Flash memory model by discussing a functional model for the Flash memory device. In Chapter 4 we will discuss electrical Flash cell models that have been published. We will then create and simulate their equivalent. In Chapter 5 we will create an industry standard electrical Flash memory model based upon our functional model and industry standard Flash cell. Finally we will conclude our discussion of Flash memory devices in Chapter 6.

2 Flash Cell and Array Architecture

We discussed the basics of memory technology in Chapter 1. Now we will further explore the Flash memory cell. A Flash cell has an FG and in this chapter we will look at its characteristics. This will give us insight in the behavior of a Flash cell. Also, we will look at some popular architectures used in Flash cell array structures.

In Section 2.1 we will look more closely at the Flash cell. Reading and writing the Flash cell will be discussed in Section 2.2. In Section 2.3 we will look at the mechanisms that make Flash writing possible. The Flash cell array architectures are discussed in Section 2.3 and in Section 2.5 we will discuss Flash cell technology and process.

We will use the Industry Standard memory Flash cell as a reference. [4] As stated by [3], Flash memory cells can be categorized in generations (Table 2.1). Our Flash reference cell is of the first generation. This generation is well documented in Flash memory literature and it does not limit our discussion of the Flash cell and memory device. Also, the more advanced Flash cell types are based upon these principles. Through the years Flash cell size has decreased and writing mechanisms and methods have improved due to research, best practices and technological improvements in CMOS process.

2.1 Flash Memory Cell

A memory device has an array of cells, which are the grouped bits. If we would use a MOS transistor to model a memory cell we need more than one transistor in case of an SRAM memory. For a Flash cell however a single MOS would suffice together with components to approach the effects of the FG. When looking at models of Flash cells in literature, it is described as a "full CMOS process in which the building blocks to get an ad hoc floating gate device are incorporated" [4]. The FG introduces a capacitive character at the MOS gate that can change the threshold voltage of this cell. The following equation describes the threshold of a regular MOS transistor [11],

$$V_{th} = K_1 - \frac{Q}{C_{ox}} \tag{2.1}$$

where K_1 is a constant depending on channel doping, gate oxide thickness, and substrate material. Q is the charge with respect to gate oxide, and C_{ox} is the capacitance of the gate oxide. The threshold voltage V_{th} can be changed by altering the charge between the gate and channel, $\frac{Q}{C_{ox}}$. In SR (saturation) region we get Equation 2.2 for I_d and in TR (triode) region we get Equation 2.3

$$I_d = K_2(V_{gs} - V_{th})^2 \tag{2.2}$$

$$I_d = K_2[(V_{gs} - V_{th})V_{ds} - V_{ds}^2] \tag{2.3}$$

where constant K_2 has units of current per $(volt)^2$ and V_{gs} is the potential between gate and source. [17]

2.1.1 Flash Cell Threshold Voltage

The threshold voltage of a Flash cell is a very important characteristic. It will define how a cell is written i.e. programmed or erased. With an industry standard cell a high threshold voltage means the cell is programmed. In Figure 2.1 a generic FG device is depicted. Its structure adheres to the ETOX structure proposed in the late eighties. [4] ETOX (EPROM Tunnel Oxide) is a FG transistor cell designed and made by Intel.

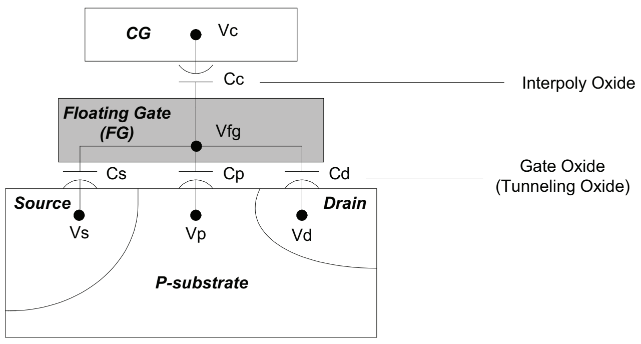


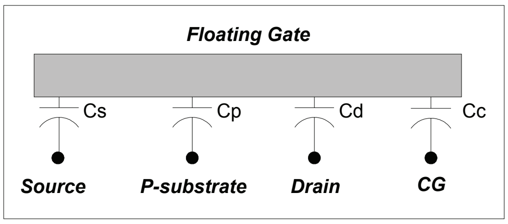
Figure 2.1: schematic cross section and electrical model of an FG device

The charge stored within an FG device (C_t) results in a potential, V_{fg}, that can be described by the following equation [4]:

$$V_{fg} = \frac{C_c}{C_t}V_c + \frac{C_s}{C_t}V_s + \frac{C_d}{C_t}V_d + \frac{C_p}{C_t}V_p + \frac{Q}{C_t} \tag{2.4}$$

where V_c,V_s, V_d, V_p are the control gate, source, drain and bulk or p-substrate respectively and Q is the charge within the FG. [4] Ct is the total capacitance which can be seen directly if one looks at the electrical model in Figure 2.2.

When we read a Flash cell based upon Figure 2.1, the bulk and source may be grounded. During a write operation, the FG will be charged or discharged to GND. We therefore change Equation 2.4 to 2.5 and refer all potentials to source. Note that these equations apply only if we read the cell. We will discuss writing mechanisms later in this chapter.



Ct = Cs + Cp + Cd + Cc

Figure 2.2: parallel capacitors in an FG device model

$$V_{fgs} = \frac{C_c}{C_t}V_{cs} + \frac{C_d}{C_t}V_{ds} + \frac{Q}{C_t} \tag{2.5}$$

We can then rewrite Equation 2.5 by defining $\frac{C_c}{C_t}$ (capacitance between control gate and FG, divided by the total capacitance) as the *coupling factor* α.

$$V_{fgs} = \alpha(V_{cs} + \frac{C_d}{C_c}V_{ds} + \frac{Q}{C_c}) \tag{2.6}$$

The characteristics of a cell depend upon the threshold voltage of the gate. Because we cannot access the FG, we need to look at the control gate, CG. The characteristics of an FG device depends on the threshold voltage (which is V_{Tfgs}) beyond which the devices starts to conduct current. We therefore need to control V_{Tfgs} through the CG. We will use the T abbreviation in the equations to describe this. At FG, we disregard $\frac{C_d}{C_c}V_{ds}$ because we are not changing any potential here. When applying a voltage to CG, we get the following threshold equation [4]:

$$V_{Tcs} = \frac{1}{\alpha}V_{Tfgs} - \frac{Q}{C_c} \tag{2.7}$$

Equation 2.6 now has a similar shape as Equation 2.1 that describes a MOS transistor, and we can use 2.7 to describe the written states of an FG device. Figure 2.3 depicts the programmed and erased states according to Equation 2.7.

Based upon Figure 2.3, an erased cell has a low threshold voltage because it has no charge trapped within the FG. This is described by Equation 2.8. For a programmed cell, the threshold voltage is high due to the charge trapped within the FG. The FG has a collection of trapped electrons, and therefore with comparison to source, or GND, it has charge Q. This charge results in an addition of potential, we can see this in Equation 2.9. We therefore get a cell with a higher threshold voltage.

$$Erased : V_{Tcs} = \frac{1}{\alpha}V_{Tfgs} - 0 = \frac{1}{\alpha}V_{Tfgs} \tag{2.8}$$

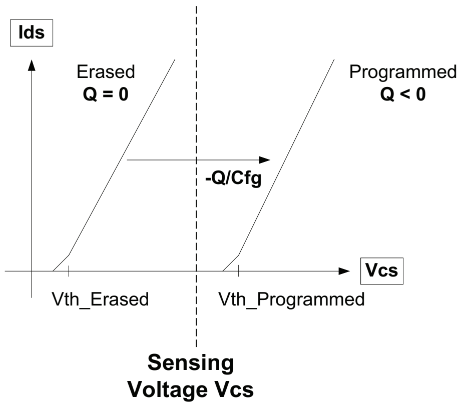


Figure 2.3: I-V characteristics of an FG device

$$Programmed: V_{Tcs} = \frac{1}{\alpha} V_{Tfgs} - \frac{Q}{C_c} \tag{2.9}$$

As for the current between drain and source of an FG transistor in SR (Equation 2.10) and TR (Equation 2.11) region, we get the following equations [4]:

$$I_{ds} = K_2 \alpha (V_{gs} - V_{th} + \frac{C_d}{C_c} V_{ds})^2 \tag{2.10}$$

where V_{gs} is the potential between gate and source.

$$I_{ds} = K_2[(V_{gs} - V_{th} V_{ds} - (\frac{C_d}{C_c} - \frac{1}{2\alpha}) V_{ds})^2] \tag{2.11}$$

The coupling factor gives us the ability to create equations that can be used to calculate the FG threshold voltage and the current of a cell. We will refer to the change of threshold level in a Flash cell due to the capacitive nature of FG as *CCM* (Capacitive Coupling Method).

2.1.2 Multi-Level Flash Cell

The demand for high density non-volatile memory devices in the computer market is rapidly growing. Alongside the design of more memory in a device, multi-level storage is an important development in current Flash design. Multi-level storage is a single memory cell that represents more than one bit. This can be done by using more than two threshold states (V_{th}). It is the most efficient way of making use of cell size. However there are tradeoffs that need to be taken into account when dealing with multi-level storage, such as a higher programming voltage needed resulting in memory cells that wear out more quickly. This results in less write cycles compared to single level Flash cells. Reading cells becomes more complex as well due the multiple levels regarding detection. Figure 2.4 depicts the different margins of a single and multi-level Flash cell. The interested reader is referred to [4] for more details.

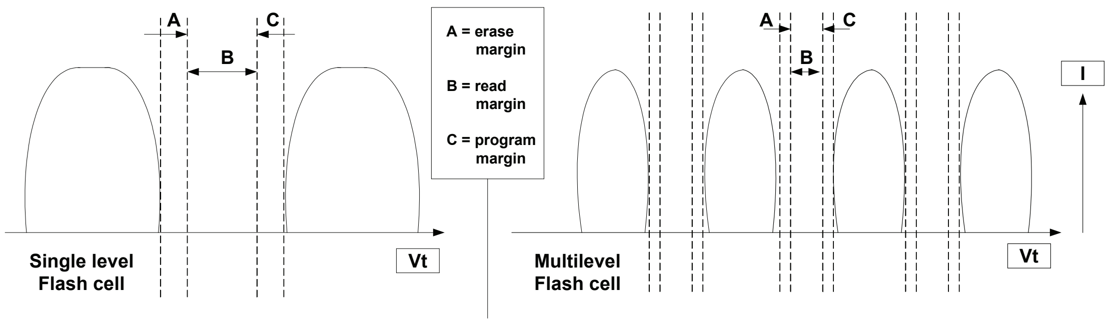


Figure 2.4: single and multi-level Flash cell margins

2.2 Flash Writing

Writing a Flash cell requires high voltages on different nodes. In this thesis we will use the first generation Flash cell as a reference for our discussion. Table 2.1 depicts 3 generations of Flash cell writing voltages for CG, Drain and Source. Currently Flash cells are far more advanced than any of these generations and have been successfully manufactured in 32nm CMOS technology with NAND cell architecture. [12]

Table 2.1: generations of the Flash memory cell

Activity	1st generation 1990-1997	2nd generation 1995-2000	3rd generation 1998 onward
Program cell	Control Gate: 12V Drain: 5V Source: GND	Control Gate: 10V Drain: 5V Source: GND	Control Gate: 8V Drain: 4V Source: GND
Erase cell	Control Gate: GND Drain: FLOAT Source: 12V	Control Gate: -8V Drain: FLOAT Source: 5V	Control Gate: -8V Drain: 8V Source: 8V

2.2.1 Erasing a Flash cell

Figure 2.5 depicts a Flash cell being erased. It also shows one cell, and a 4 bit block of cells. To illustrate the writing process of an array of cells we will use the NOR architecture. We will discuss NOR and other Flash cell architectures in more detail in Section 2.5.

A Flash memory array is divided in blocks. All cells in a NOR array block are erased simultaneously. This is depicted by the circles. The control gates are set to GND and a high voltage (Vdd) is applied to all sources. It is common that a read-check is performed after an erase operation to make sure all cells are erased in the entire block. Table 2.1 shows the voltage values of erasing a first generation cell. Erase time typically lies between 100ms - 1s. [4]

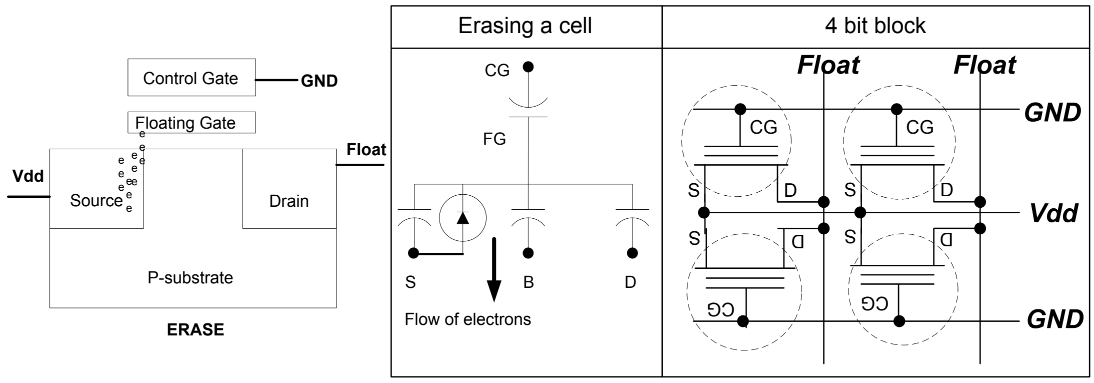


Figure 2.5: erasing a Flash cell

2.2.2 Programming a Flash Cell

Figure 2.6 depicts a Flash cell being programmed. Programming a Flash cell happens one bit at a time for a single block. Different blocks can therefore be programmed simultaneously. When storing one bit at a time in each block n blocks are needed to represent an n-bit word. Programming time for a cell ranges between 1 - 10us. [4]

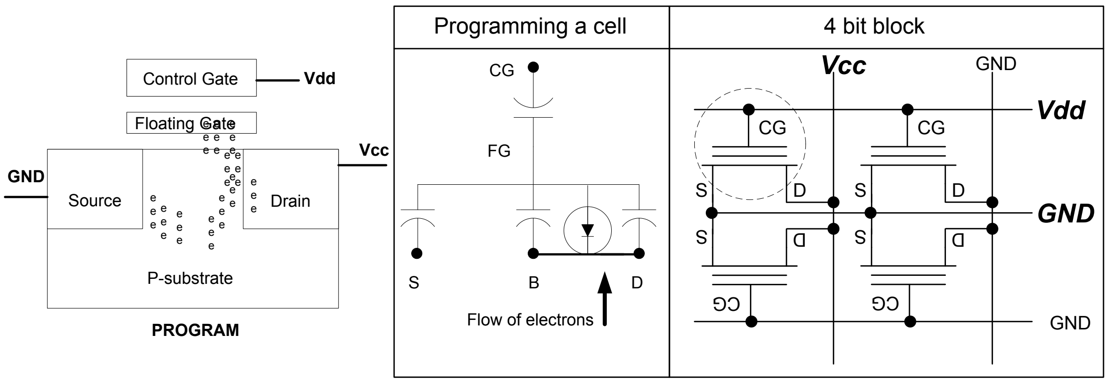


Figure 2.6: programming a Flash cell

2.2.3 Reading a Flash cell

Access time for reading ranges between 50 - 100ns. [18] Figure 2.7 depicts this situation. A programmed Flash cell has a higher threshold level than an erased cell. This means that the logical state can be regarded as logical 0, i.e. cutoff mode for the cell therefore no current flows. As for an erased cell, the opposite is true. Figure 2.9 gives a simplified illustration of how the blocks in a NOR array can be represented as words. Note that each array architecture has its own bit representation structure of a word. Logically we only need to distinguish the written modes and convert them to a logical 1 or 0 when sensing the states.

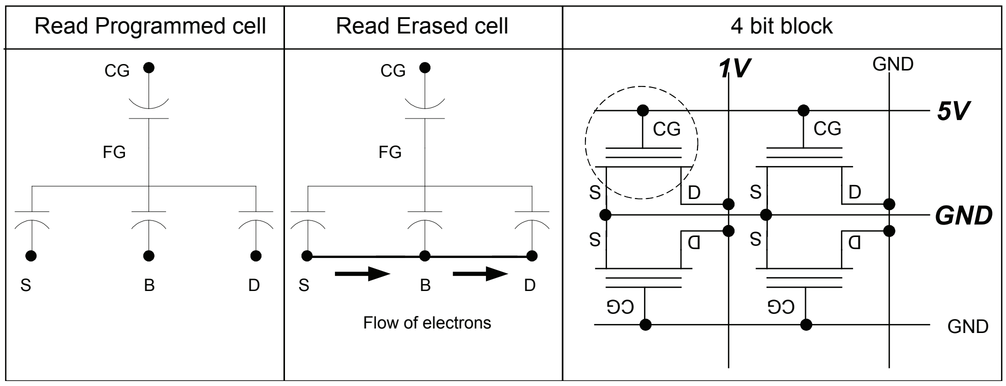


Figure 2.7: reading a Flash cell

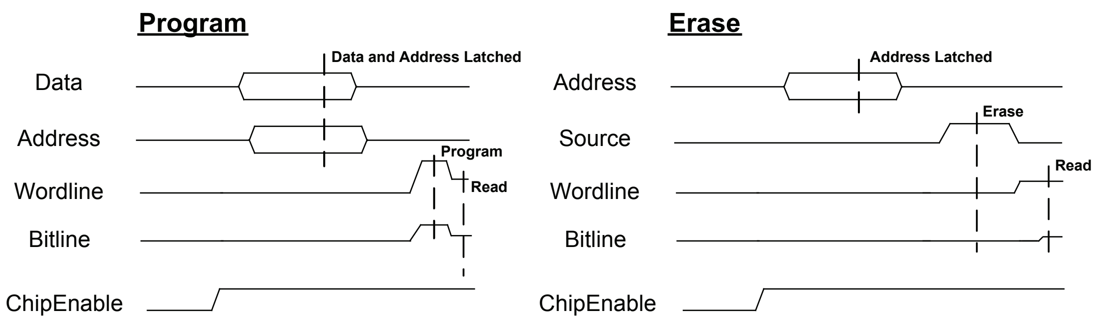


Figure 2.8: timing diagram writing process

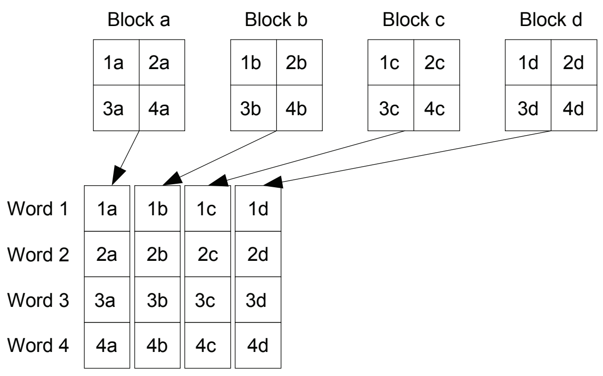


Figure 2.9: blocks and corresponding words of NOR

When writing Flash devices, we need to keep reliability in mind. An FG cell is very delicate and disturbances may occur with an adjacent cell due to writing with high voltages. To minimize these effects, programming only one cell at a time in a NOR block is more responsible. When reading a cell with 1V, disturbances are kept low. If reliability issues improve, a more parallel programming process could become possible in a block. Current generations have very low programming voltages and use select transistors to keep disturbances in blocks and adjacent cells to a minimum.

2.2.4 Flash Writing Management

Flash memories generally have their own memory management. This combination of logic and memory (e.g. tables) simplifies the usage of the Flash memory device and ensures a correct writing process. The writing process or algorithm can be depicted in a simplified flow chart in Figure 2.10.

For programming, management stores address and data, and sets a counter variable for this specific process. To minimize unnecessary cell activity the bits are checked at the specified address. If the cells are not programmed then these cells are targeted for programming. If the counter exceeds the maximum number of program tries, we encounter a programming error. When a comparison needs to be made, a read operation of the cells is performed.

For erasing this process is slightly different. At first no comparison is made, the entire block is erased and checked afterwards. This means that already erased cells also undergo this process. The management therefore checks if there is not any sign of over erasure of cells. Checking for programming or erasing errors can for one indicate a problem with the FG.

2.3 Flash Writing Mechanisms

FG devices use writing mechanisms for programming and erasing a cell. Two commonly used mechanisms for FG devices are *FN Tunneling* (Fowler-Nordheim Tunneling) and *CHEI* (Channel Hot Electron Injection). [4] Tunneling mechanisms generally use less current than injection mechanisms. It is therefore more advantageous to use a tunneling mechanism. However, the tunneling mechanism is relatively slower than the injection mechanism. FN Tunneling is used for programing and erasing whereas CHEI is mainly used for programming.

2.3.1 CHEI

CHEI is a write mechanism that is also called "Hot Carrier Injection". It is a mechanism where the electron collects enough energy to overcome a potential barrier. The electrons that flow through the channel are 'heated' by a large lateral electric field between drain and source. This means that the electrons are given a boost via a bias potential at the drain, and therefore gain enough kinetic energy to pass through the oxide barrier

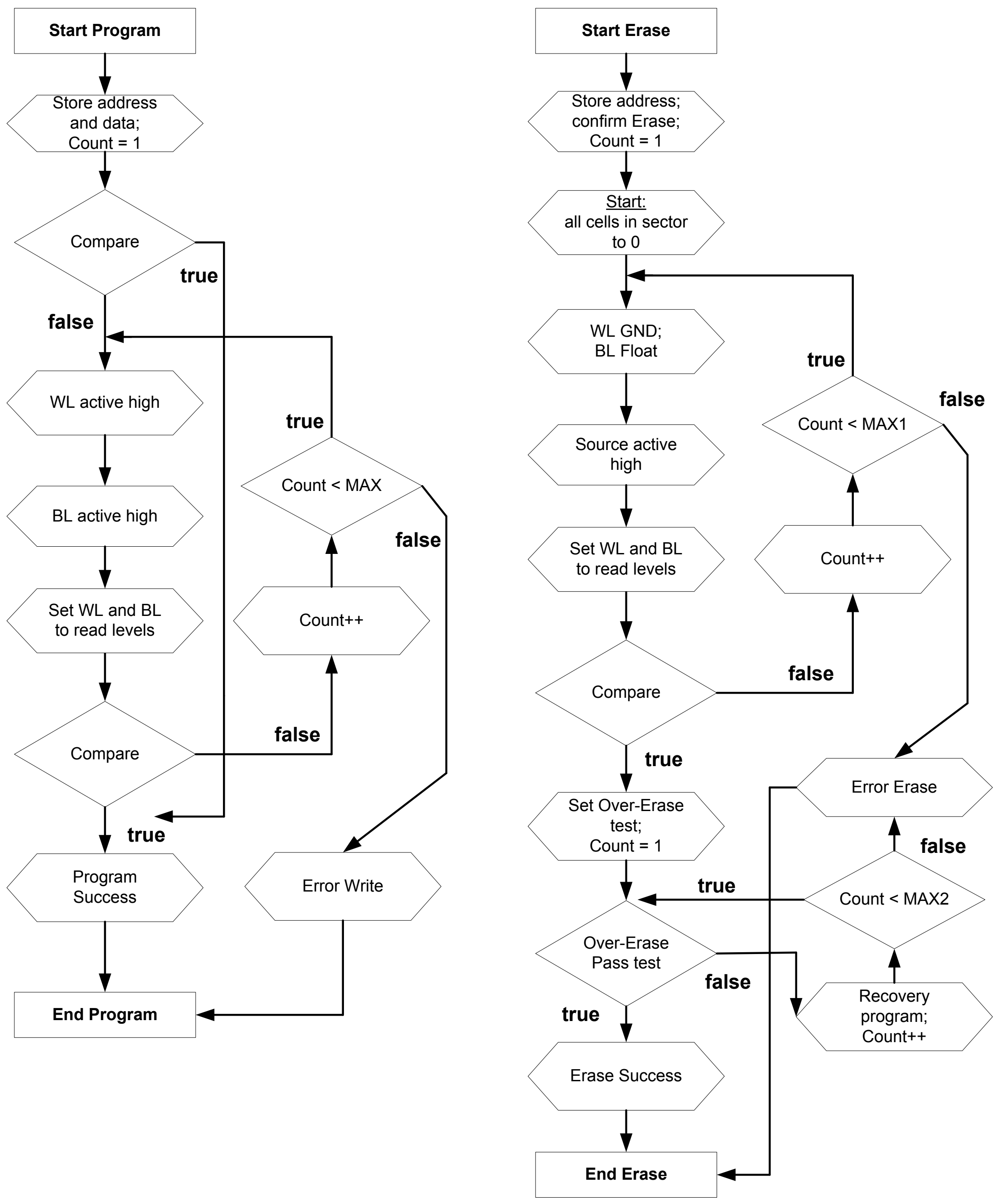


Figure 2.10: writing method flow chart

towards the FG. The hot electrons travel from source towards the drain and near the drain region they have energy to overcome the SiO2 energy barrier. They then get injected into the floating gate due to the electric field (E_{ox}) across the channel. [11]

CHEI is difficult to describe mathematically due to its many unknown physical parameters and character. However, a model called the "Lucky Electron" model has been able to describe its nature analytically. This model has specific criteria that includes gaining sufficient kinetic energy in an electron without losing energy due to collisions. An equation can be described that determines the gate current in Equation 2.12. [10]

$$I_{CHE} = \alpha_{ox} I_{sub} exp\frac{-\beta_{ox}}{E_{ox}} \tag{2.12}$$

where α_{ox} and β_{ox} are fitting parameters, E_{ox} is the electric field into the tunnel dielectric and I_{sub} is the substrate current.

$$I_{sub} = I_{DS}\frac{a_i}{b_i} V_{sat} exp\frac{-\beta_i}{V_{sat}} \tag{2.13}$$

where I_{DS} is the channel current, a_i and b_i are impact ionization coefficients, and V_{sat} is the saturation potential. I_{DS} can be determined by an equation that takes channel width W, channel length L, inversion charge, surface potential, and carriers effective mobility (which is a function of the linear and quadratic mobility attenuation factors, saturation voltage and carrier velocity saturation) into account.

2.3.2 FN Tunneling

FN Tunneling tunnels electrons through a layer of oxide (of an exact rounded triangular barrier) and is an important tunneling mechanism for thin oxide barriers. Compared to CHEI it can achieve the needed effect with less power. It is a form of quantum tunneling, a phenomenon where particles have been given sufficient energy to break through a barrier (an energy state) due to an electrostatic field. The electric field across the oxide layers determine the measure of electrical current density through the oxide layer. Typically the oxide thickness is about 10 nm from FG to P-substrate. A distance of less than 6 nm will result in insulation failure, thus a bad functioning FG cell. A simplified version of the Fowler-Nordheim equation describing the tunneling current is shown in Equation 2.14. [14] [20]

$$I_{FN} = A(E_{ox})^2 exp\frac{-B}{E_{ox}} \tag{2.14}$$

$$A = \frac{q^3 m}{8\pi h \phi_b m_*} \tag{2.15}$$

$$E_{ox} = \frac{V_{app} - V_{fb}}{t_{ox}} \tag{2.16}$$

$$B = 4(2m^8)^x \frac{\phi_b^y}{3h_* q} \tag{2.17}$$

where h is Planck's constant, ϕ_b is 3.2 eV, m is the mass of a free electron, m_* is the effective mass of an electron in the SiO2 gap (0.26 m), h_* is 0.16 h, x is 0.5, y is 1.5, V_{app} is the applied voltage across the oxide, V_{fb} is the flat band voltage and t_{ox} is the thickness of the oxide.

2.4 Flash Array Architectures

The architecture used in a block of cells will determine the characteristics of a memory device, i.e. block size and access time. These architectures fundamentally differ in group wise connection of CG, Drain and Source, and also in writing mechanism and voltage. We will look at 4 of the most widely used array architectures, namely NOR, NAND, AND and DINOR (Divided Bit line NOR).

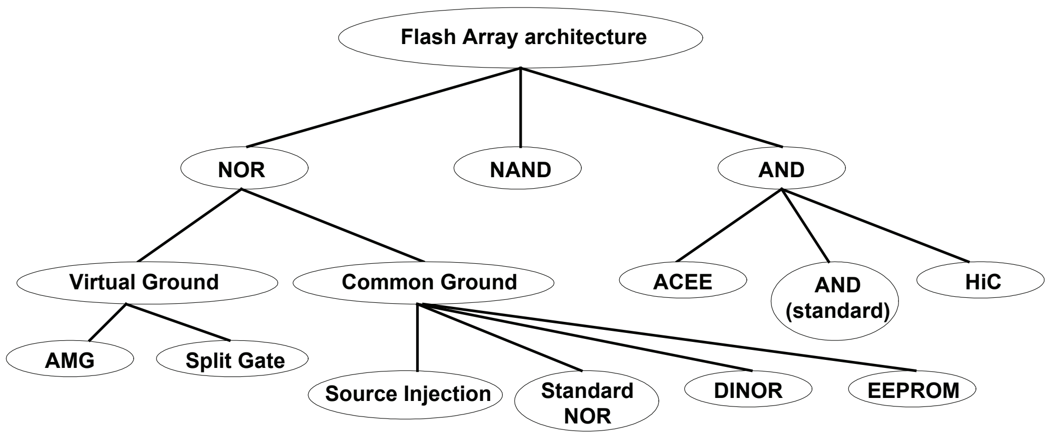


Figure 2.11: Flash array architecture overview

In Table 2.2 an overview is given of the 4 popular Flash array architectures and manufacturers that have produced Flash memory devices based on them.

Table 2.2: Flash memory manufacturers in the late nineties

NOR	NAND	AND	DINOR
Toshiba	Toshiba	Hitachi	Hitachi
Mitsubishi	National	Mitsubishi	Mitsubishi
AMD	AMD		Motorola
Fujitsu	Fujitsu		
Samsung	Samsung		
Intel			
Atmel			

The NOR architecture we will discuss is also called *standard NOR*, which is a common ground architecture. This term is used for a structure where every two FG cells have a common source and drain. Figure 2.6 depicts this structure. The common source is a diffusion line connected to ground via a dedicated metal wire every 16 bit lines. A virtual ground architecture is quite similar to a common ground architecture, except for a dedicated metal wire every 64 bit lines.

2.4.1 NOR Architecture

The NOR (common ground) architecture is one where each cell can be read directly, i.e. having to pass through any other cell which are cells in serial. Similar to the logical OR gate, NOR has an 'OR' setup regarding its cells being read, thus its name. We can see in the figure that the sources of 2 cells are connected to each other. Figure 2.5 and 2.6 have shown this characteristic. Programming is generally done via CHEI and erasing via FN Tunneling. Figure 2.12 depicts the NOR architecture. Table 2.3 depicts the levels of reading and writing the NOR cell.

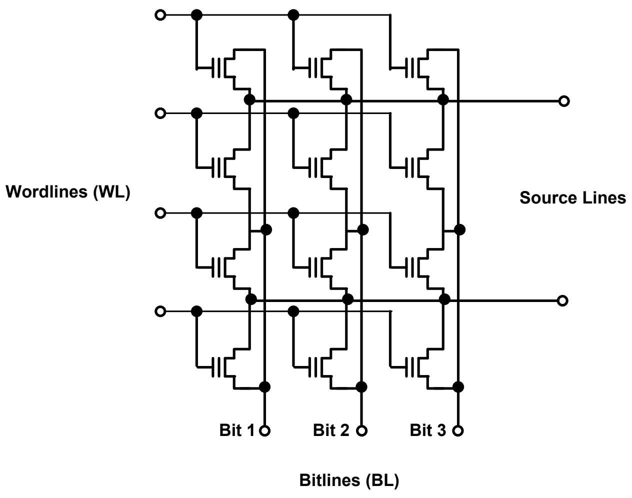


Figure 2.12: NOR architecture

Table 2.3: NOR read and write voltages; [4]

Mode	CG	Drain	Source
Read	5V	1V	GND
Program	12V	5V	GND
Erase	GND	float	12V

2.4.2 NAND Architecture

The NAND architecture has a different block structure than NOR. It has a serial structure directly connecting the cells from source to drain. It is therefore possible to place the cells closer to one another. A block is generally formed by groups of 16 serially connected cells with 2 select transistors (the BL select transistor (BSL) and the ground select transistor (GSL) in Figure 2.13) in each group. If a cell is selected, the other cells must be set to read-through. When reading a cell in this architecture, access time will be slower than reading the cell more directly as with the NOR architecture. Also, due to the read-through nature, a higher voltage is needed for write operations in each block.

An erased cell has a negative threshold voltage in the NAND architecture. A programmed cell has a positive threshold voltage. The CG of the selected cell to be read is set to 0V and the others are set to read through, which is above the programmed cell threshold. To create the negative threshold voltage in a cell (erased cell), the bulk is set to 20V, the CG to 0V and the CGs of unselected cells to 0V. For a positive threshold (programming) the bulk is set to 0V, the CG to 20V and the other CGs are set to 10V. [4] Programming and erasing is generally done via FN Tunneling. Figure 2.13 depicts the NAND architecture. Table 2.4 gives an overview of the levels of reading and writing the NAND cell.

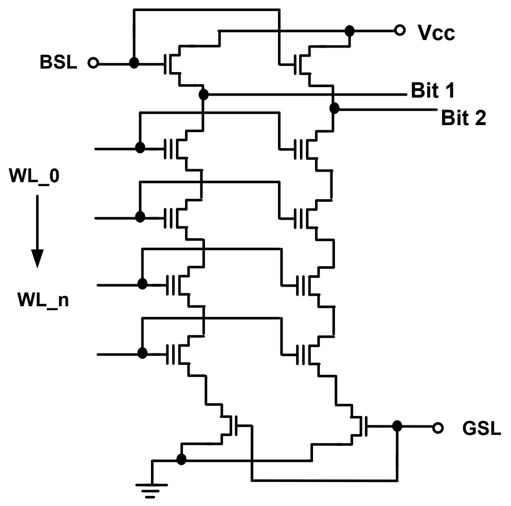


Figure 2.13: NAND architecture

Table 2.4: NAND read and write voltages; [4]

Mode	CG	unselected CG	Bulk	BSL	GSL
Read	GND	5V	GND	5V	5V
Program	20V	10V	GND	5V	GND
Erase	GND	GND	20V	float	float

The NAND architecture has approximately 40 percent more cells on an area compared to NOR. Figure 2.14 depicts an illustration of cell placement. NAND is generally used for mass data storage and is a slower memory to write.

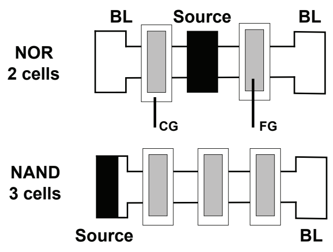


Figure 2.14: NOR and NAND cell placement comparison; simple geometrical view

2.4.3 AND Architecture

The AND architecture is more or less a combination of NOR and NAND. It has better access times than NAND due to its structure. Also, the cells can be placed closer to each other than in the NOR architecture. The AND architecture adopts the reverse convention of cell programming and erasing, i.e. the AND developers have agreed that a low threshold voltage corresponds to a programmed cell and a high threshold to an erased cell. [4] As for programming and erasing, FN Tunneling is commonly used. Figure 2.15 depicts the AND architecture. Table 2.5 depicts the levels of reading and writing the AND cell.

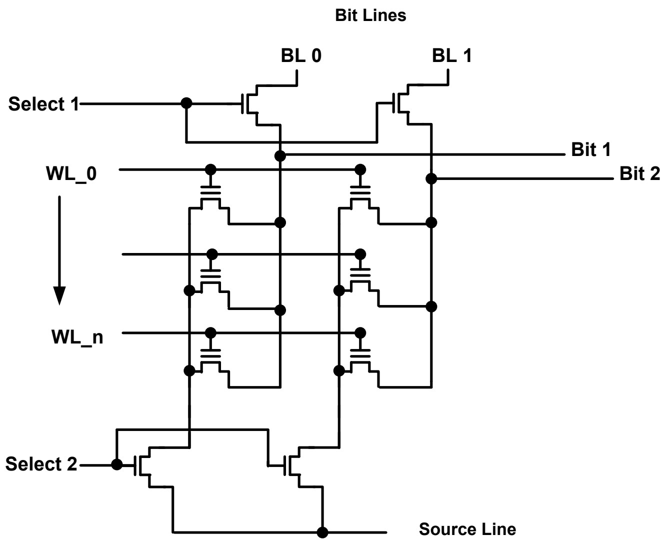


Figure 2.15: AND architecture

Table 2.5: AND read and write voltages; [4]

Mode	CG	BL	Source
Read	5V	1V	GND
Program	-8V	6V	Float
Erase	10V	Float	-8V

2.4.4 DINOR Architecture

The DINOR architecture is similar to the AND architecture. The major difference between them is that DINOR has a common ground connection without a select transistor. Like an AND array, this architectures benefits from better access time than NOR, with even less area than AND due to the common ground connection. This architecture also has a reversed convention for programming and erasing and FN Tunneling is generally used for programming and erasing. Figure 2.16 depicts the DINOR architecture. The AND read and write voltages apply to this architecture as well.

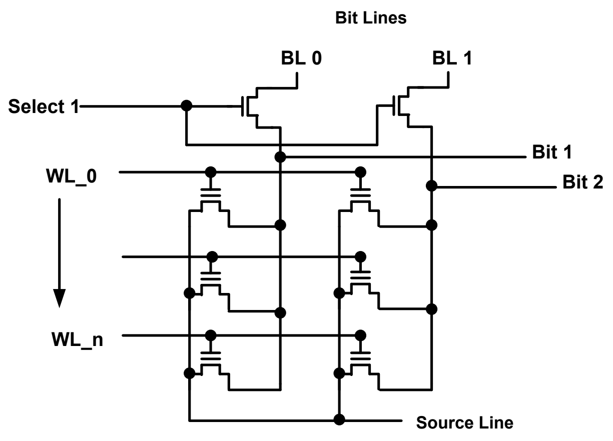


Figure 2.16: DINOR architecture

2.4.5 Array Architecture Overview

In the previous subsections we have seen that an array architecture is accessed either in parallel or serially. A combination can also be applied, i.e. AND. This can even be complemented by including specific structures (e.g. a tree structure) with select transistors in order to make the design faster. This eventually determines the size of the array, the read and write time. Cell process technology also has a great influence on performance. Manufacturers will use a specific combination of these to create a device.

We will now look at some figures of merit in Table 2.6 describing the previous Flash array architectures. [4] These figures represent *cell size*, *process complexity* (this is directly linked to the process of manufacturing the cell), *array efficiency* (a correlation of cell and array complexity to the product implementation and yield), *general purpose application* (both high performance and data storage purpose), and *high voltage requirements*. This last figure describes the level of complexity of the CMOS process needed that enables it to withstand high voltages. Note that these figures range from 1 to 3 with 1 representing the best overall performance. There are many more figures of merit, this selection is shown to give an indication of the many subtleties that impact Flash array performance.

Table 2.6: figures of merit of popular Flash array architectures

	NOR	NAND	AND	DINOR
Cell size	3	1	2	1
Process Complexity	2	2	3	3
Array Efficiency	2	2	3	3
General Purpose	1	3	1	1
High Voltage Requirements	2	3	3	3
Total:	10	11	12	11

2.5 Flash Process Technology

As discussed, Flash process is full CMOS with specific methodologies to create an FG. This means that standard CMOS technology steps with additional complex features have to be taken. [4] CMOS circuits are fabricated on and in a silicon wafer. This wafer is doped with donor atoms. Phosphorus is used to create an n-type wafer and Boron for a p-type wafer. By exposing the wafer to air the SiO2 layer is grown, i.e. Si + O2. This can be done a lot faster by exposing the silicon to hot steam. However, a side effect is that a hydrogen impurity occurs in the silicon. Figure 2.17 depicts a simplified air grown process. It also shows the Field Oxide (FOX) regions that are grown to isolate active p-well and n-well regions. This technique is also called LOCOS (Local Oxidation of Silicon). [2] This figure shows the basics of circuit preparation.

Besides the usual requirements of standard CMOS in terms of access time and low operation voltage, a high voltage is needed for writing operations and this involves FN Tunneling or Injection. Amongst others, it is required that writing takes place without degrading data retention. All these requirements make Flash technology very difficult to master. Flash technology process can be divided in a front end and back end shown in Table 2.7.

Isolation is needed to prevent parasitic leakage current between neighboring cells. The cells must sustain high voltages (of greater than 10V). This is especially needed in areas where the circuitry is very dense (e.g. row decoding area). After well and channel

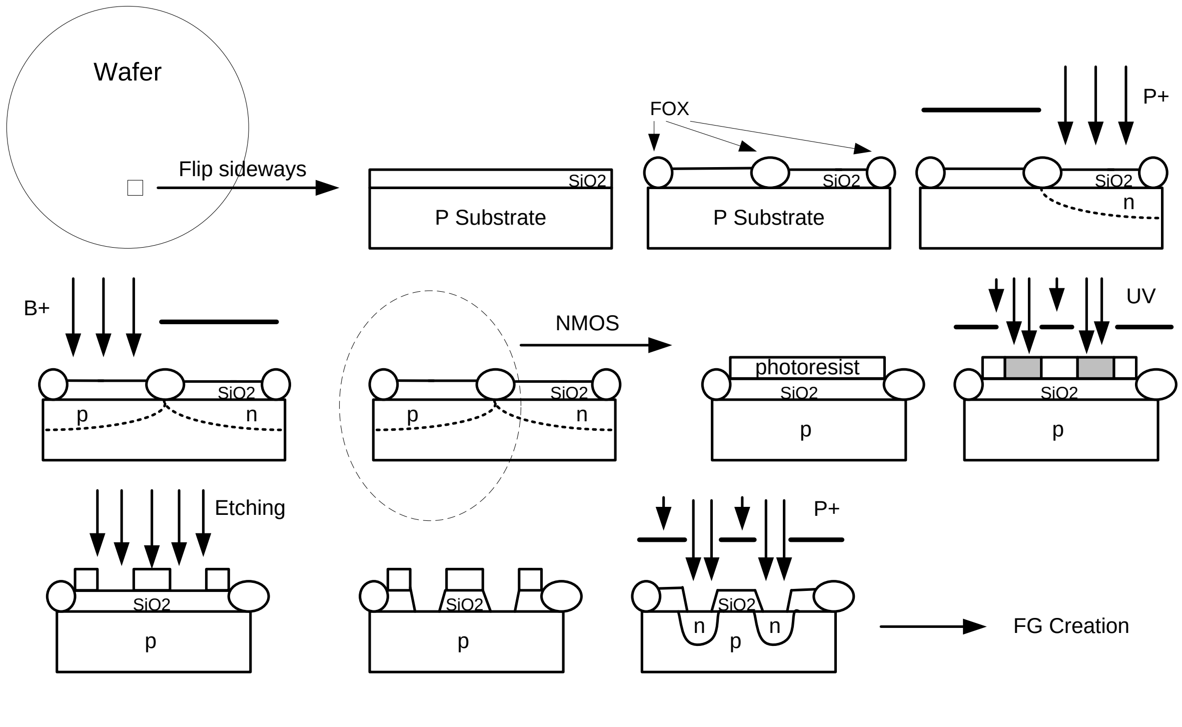


Figure 2.17: silicon wafer; front end: simplified process of circuit preparation

Table 2.7: basic process flow of Flash technology; [4]

Front End	Back End
1) Isolation	5) Interlevel Dielectric
2) Well and Channel doping	6) Interconnections
3) Cell structure definition	7) Passivation
4) Transistor definition	

doping to acquire the different MOS parts, cell structure is thoroughly defined in order to create reliable cells. After these steps, the FG is added to the circuit. Figure 2.17 ends with this step and afterwards transistor definition is completed. In order to reduce intrinsic charge loss between the channel and FG, the phosphorus content in between them is carefully concentrated. This is the interlevel dielectric part of the process. After these steps, all the required contacts are made and surface planarization is applied. The last step is to protect the circuit against contamination compounds by passivation. After this step the floating gate is added to the MOS transistor. The interested reader is referred to [4] for more details of these last steps. Note that each manufacturer has its own methodologies. We will discuss Flash memory as a whole from a functional point of view in the next chapter.

3 Flash Memory Device

In the previous chapter we have seen the Flash cell and its structure in arrays in order to give us a better understanding of its nature due to the FG characteristics and cell array architecture. This chapter will give us a functional level overview of a Flash memory device. From this functional model we will develop an electrical Flash memory device model in the next chapters.

In Section 3.1 of this chapter we will look at memory modeling in general, based upon the traditional RAM memory overview. This memory type was thoroughly described in [16]. Section 3.2 will give us an overview of the functional Flash memory model. This will enable us to construct a simplified electrical model based upon these functional blocks, without any detailed design from specific manufacturers.

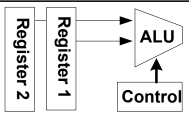
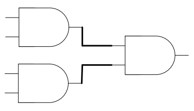
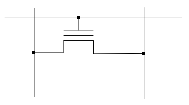
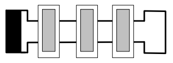
3.1 Memory Modeling

Models enable us to simplify and structure an entity and its environment. Structuring further adds simplification because a specific model then only targets the relevant aspects and phenomena for discussion. We can look at a memory in terms of what it should do (behavioral), in terms of blocks that make sure its main function is accomplished (functional), in terms of how the circuit is connected with electrical components (electrical) and how these components are created with its specific dimensions and technology (geometrical).

By looking at a functional model of a memory, we can logically determine and verify the behavior of the system. At this level one does not have to get bogged down by electrical details and this abstraction enables a technology-independent and general approach. However, going more into details gives one the ability to create accurate simulations to verify the levels above. In this thesis, we will therefore focus upon the functional and use this to create electrical model. Figure 3.1 gives an overview of the modeling levels described in [16].

A traditional memory is the RAM device. Figure 3.2 depicts the functional model of this memory device. Block A functions as a latch and targets the appropriate rows and columns. Block D is set up in a specific structure to represent the words of memory. Block F amplifies the bit(s) from the array in order to represent the stored bit(s) to the outside world. Block G functions as gateway to the outside world by either storing bit(s) for storage or preparing bit(s) for reading. Block E functions as read and write control.

Increasing level detail

Model type	Description	Example
Behavioral Model	Specification of a system. A description of what needs to happen and how. Due to invisibility of internals it is also called a **black-box** model.	**for (i=0; i<k; i++)** **{** **f = r - (t << i);** **}**
Functional Model	Functional description of a system. Assumptions are made of the internal structure. Also called a **grey-box** model.	
Logical Model	Logical description of a system at the level of Logical gates. Basically a sub-level of the functional description.	
Electrical Model	Electrical description of a system with the knowledge of internal structure, i.e. interconnections of transistors. Also called a **white-box** model.	
Geometrical Model	Model with specifics of the layout of the circuit, i.e. doping levels and physical distances such as transistor channel width.	 Figure 2.15

Increasing level of abstraction

Figure 3.1: abstract modeling levels; [16]

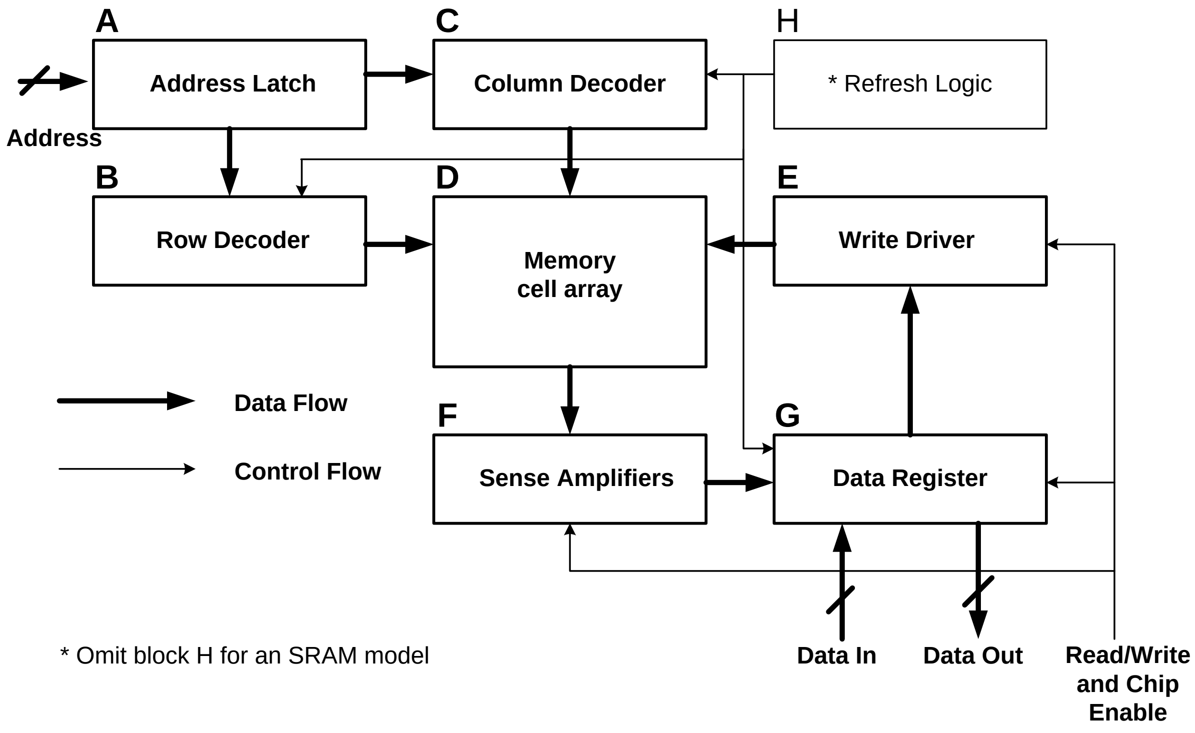


Figure 3.2: functional (S/D)RAM model; [16]

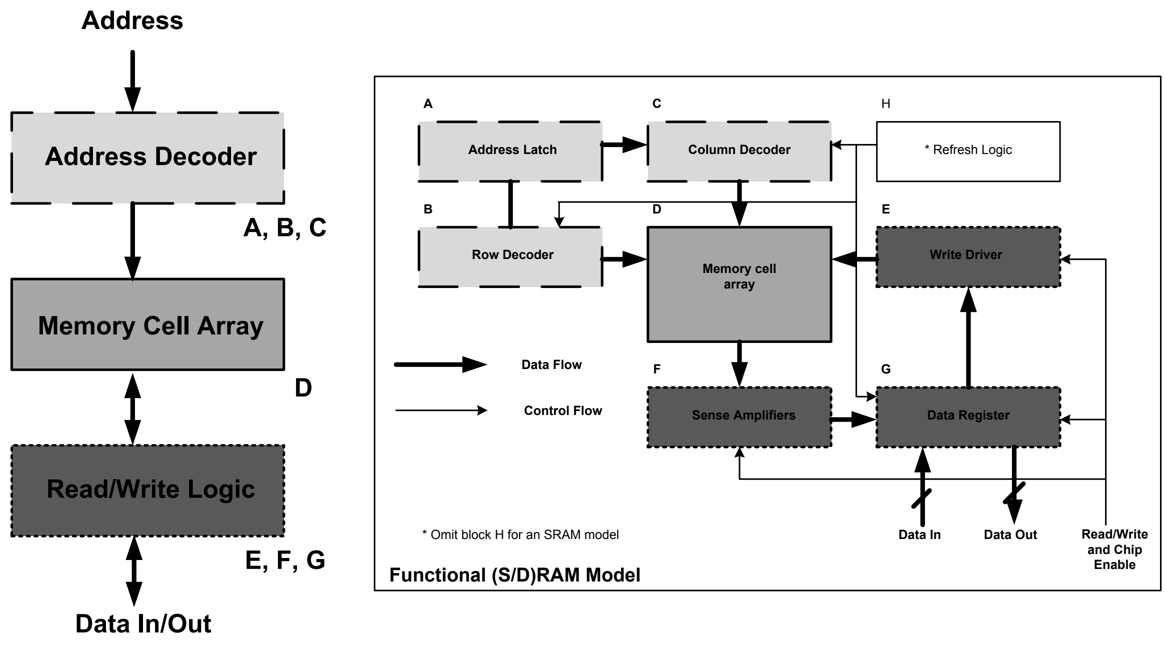


Figure 3.3: reduced functional (S/D)RAM model; [16]

3.2 Functional Flash Memory Device

To transform the functional RAM model in Figure 3.2 into a Flash model, we need to take the nature of a Flash cell into account. To read a Flash cell we do not yet need any high voltages but this changes as soon as we want to write a Flash cell. We also need to take source voltages into account when erasing. This gives the need of an extra decoder or source switch. The write driver needs to be a unit that enables write management, we discussed this in Section 2.1.4. Based upon the flow chart, we also need to compare a cell with the preferred state (hardwired) and have counters that keep track of the number of write tries (more than n write tries may reveal an overwritten cell). There are more considerations one can take to model a detailed Flash memory device, but we will stop here. We can simplify Figure 3.2 to a reduced form shown in 3.3. The functional Flash model (*FFM*) is depicted in Figure 3.4. By modeling it this way, it looks slightly like the functional RAM model proposed by [16].

An architectural block diagram of a Flash memory device described in [4] is similar to our FFM aside from several detailed blocks such as a command interpreter (preparing the device to execute a decoded instruction) and spare logic. We will use this FFM to construct an even more basic model that will form the basis of our electrical model.

If we look at the FFM more closely, we see that apart from the changes in data flow and unit functions we have discussed, there is also a difference in the column decoder.

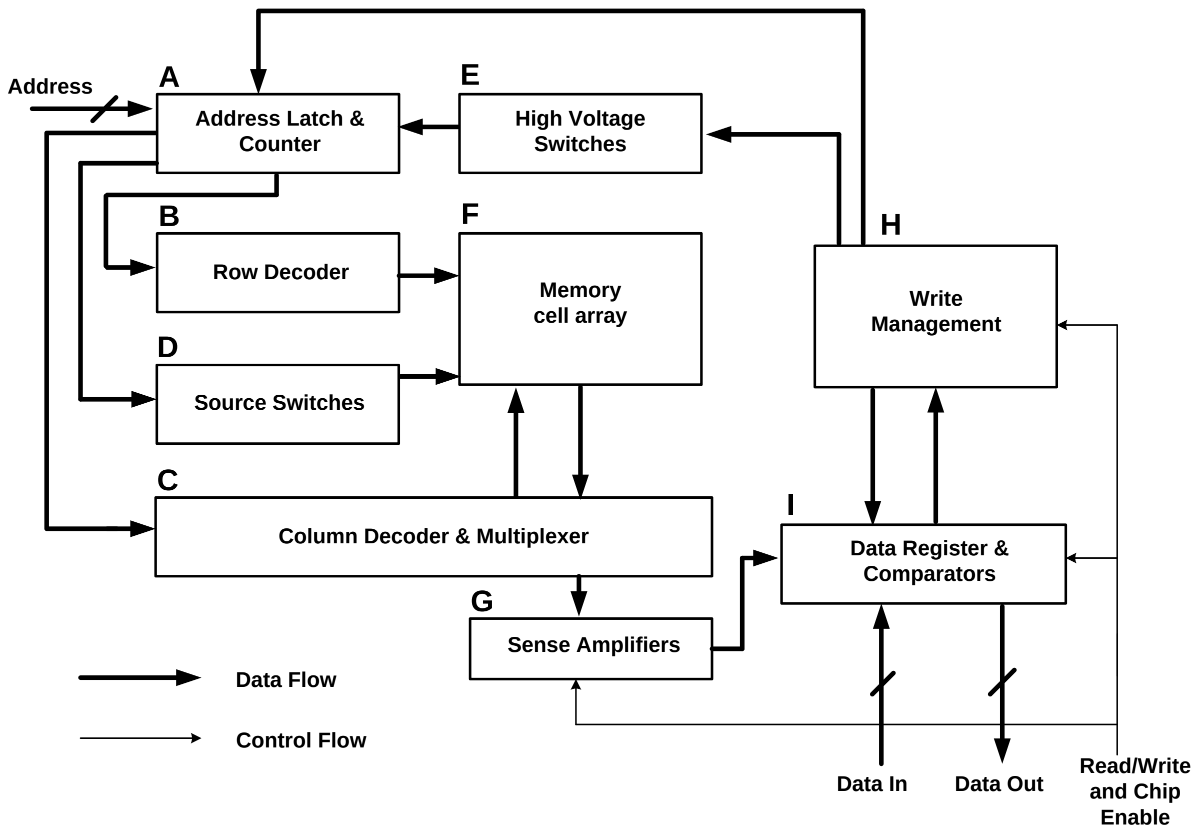


Figure 3.4: functional Flash model (*FFM*)

In the FFM it is placed below the array of cells and it has a multiplexer. We will see in Section 3.2.2 that the sense amplifier and column decoder are placed close to each other. A reduced model of FFM (*RFFM*) is depicted in Figure 3.5. By combining the internal units we see that we end up with a reduced functional model that looks exactly the same as Figure 3.3.

3.2.1 Reading and Writing in the FFM

The following steps are required for **reading** data [4] in the FFM in Figure 3.4:

1) 'Read/Write' and 'Chip Enable' are set
2) Data registers (I) are set to 'Read data'
3) Write Management (H) sets Address Latch (A),
Row decoder (B) targets CG,
Column Decoder (C) targets BL,
Source Switch (D) targets source
4) Column Multiplexer (C) sends data to Sense Amplifiers (G)
5) Sense Amplifiers (G) sends data to Data registers (I)

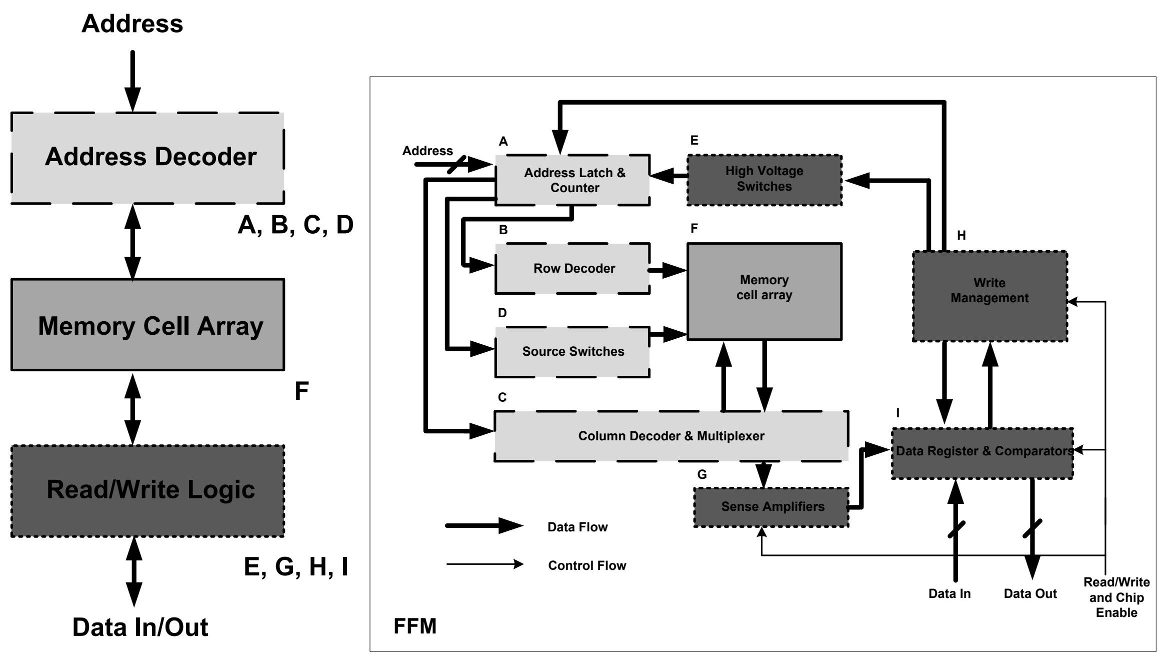


Figure 3.5: reduced functional Flash model (*RFFM*)

6) Data register (I) provides Data Out

These steps are needed for **writing** data [4] in the FFM:

1) 'Read/Write' and 'Chip Enable' are set
2) Data registers (I) receive 'Data In'
3) Write Management (H) sets Address Latch (A),
Write Management (H) sets Voltage Switches (E),
Row decoder (B) targets CG,
Column Decoder (C) targets BL,
Source Switch (D) targets source
4) Column Multiplexer (C) sends data to Sense Amplifiers (G)
5) Sense Amplifiers (G) sends data to Data registers (I)
6) Write Management (H) sets Comparators (I),
IF (comparison with reference cell) goto step 7
ELSE (IF (count $\geq$ MAX) goto step 3 ELSE overwritten)
7) Data register (I) provides Data Out

3.2.2 Basic Functional Flash Model

We will next create a simple model that adheres to the layout of RFFM. Let us take a closer look at the FFM. If we leave out all the blocks except for the cell array decoders, cell array and the sense amplifier, we still have a model that covers a working core to represent a functional Flash model. Also, we still have a subset of the RFFM model. Figure 3.6 depicts this model.

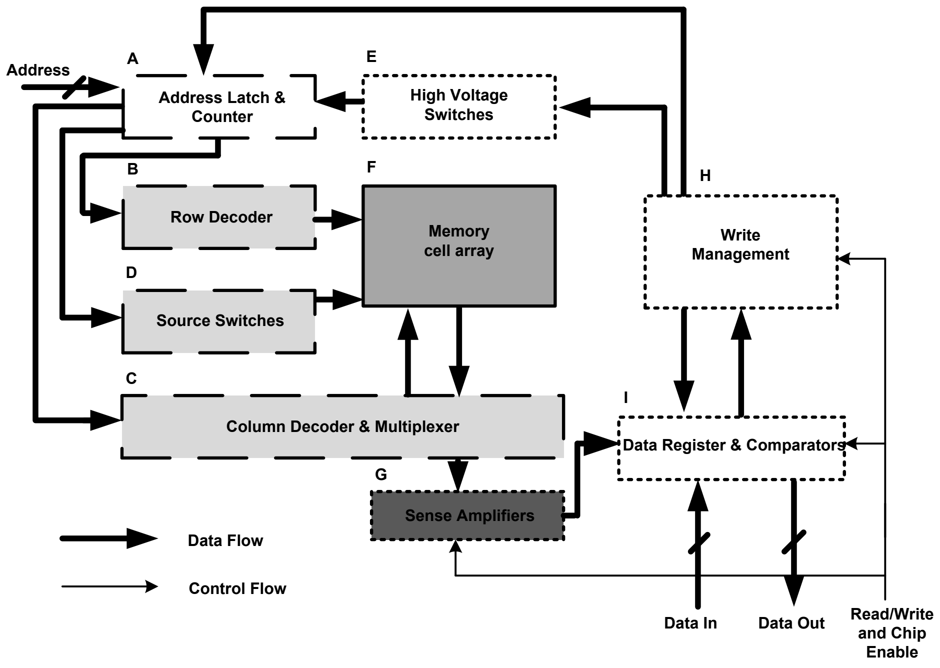


Figure 3.6: simplified FFM that adheres to RFFM

Figure 3.7 depicts a simplified model based upon the model in Figure 3.6. We will use this model to construct an electrical model later on. We will refer to this model as the Basic Functional Flash Model (*BFFM*).

3.2.3 Flash Memory Device Model

Let us look at an actual Flash memory device model. We will target a simplified view of a 64 Mb Flash memory device model fabricated in 0.18 μm CMOS technology with NOR type architecture implemented. [11] In Figure 3.8 a cross section of a Flash cell is depicted that is used in the sectors of the Flash memory device we will discuss. A programmed Flash cell sinks less current than an erased cell due to the higher threshold voltage. Reading Flash cells is done by converting current at V_d (drain) to voltage and by using a voltage comparator with reference cell. Memory designers have to pay special attention to parasitic components with regard to read speed. Also, one must avoid electrical stress on un-addressed cells. Cells in this model are programmed with CHEI by injecting electrons in FG. V_{cg} (control gate) is 10V and Vd 4.5V. Nowadays Flash memory devices (e.g. MP3 players) with a power supply of 0.9V to 1.6V can achieve appropriate values for programming. This means that programming currents have decreased from 1 mA to the range of 100 μA. Erasing a cell is done by setting V_{cg}

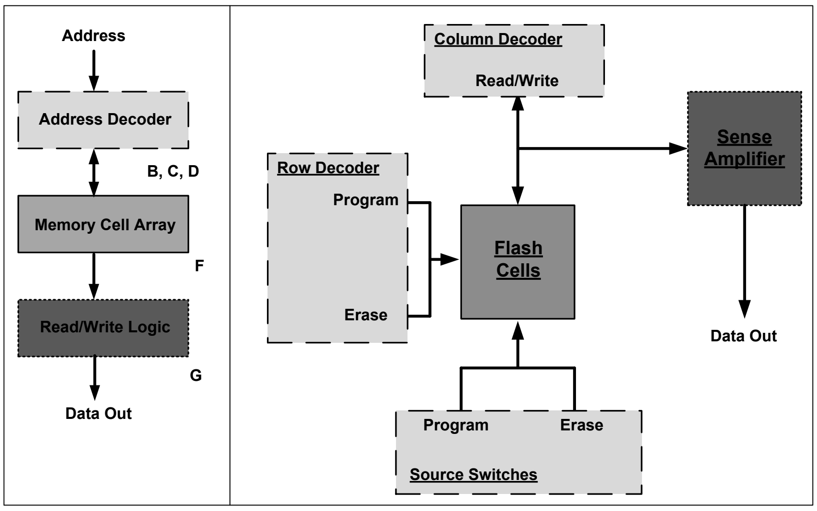


Figure 3.7: basic functional Flash model (BFFM)

to -8 V, V_d and V_s to 5 V. Here FN tunneling is used to remove charge from FG.

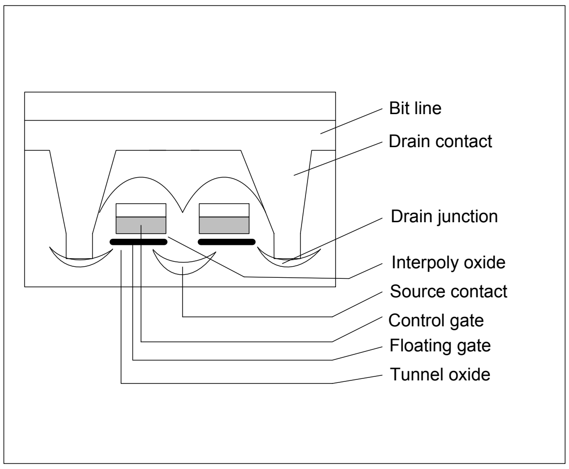


Figure 3.8: Flash cell cross section; [11]

In Figure 3.9 each sector of 1Mb is separated by a local row and column decoder. The sectors are implemented to avoid as much cross reference as possible. The local row and column lines target specific cells in a sector. All sectors are directly connected to the source switches and drain pumps required for writing Flash cells. Apart from the pumps, array of cells, decoders and reference matrix we see storage components and control and test logic to realize the required command. The ATD (Address Transition Detector) is a unit that responds to address variations and fires a clocked process to read the required cells. These steps are also described in Section 3.2.1. The CUI (Command User

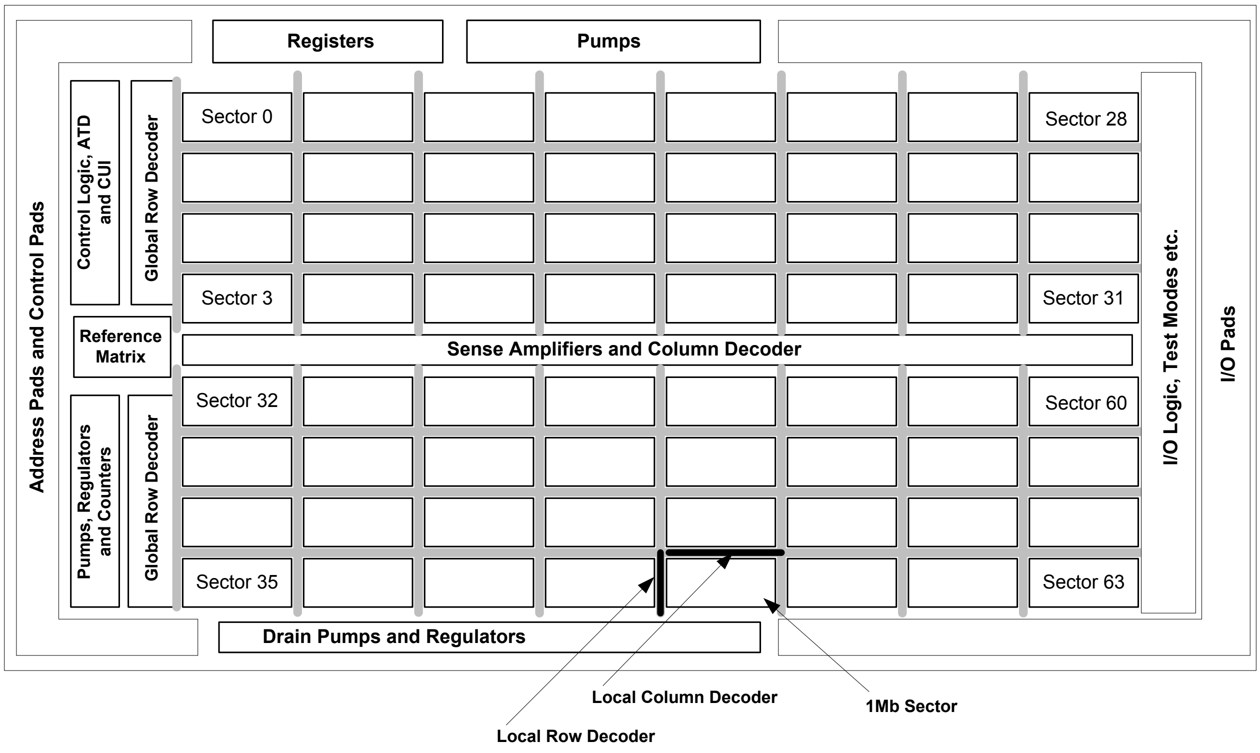


Figure 3.9: layout of a 64 Mb Flash memory device; [11]

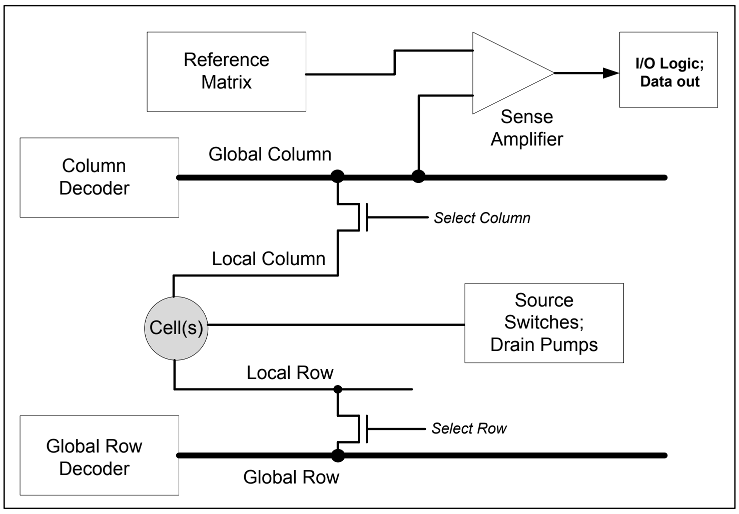


Figure 3.10: simplified layout of a sector from Figure 3.9

Interface) unit acts as a Finite State Machine (*FSM*) in order to execute the required action. There are several other FSM units that are not depicted here to keep this model simple. Apart from strict guidelines in creating a memory device with a chosen technology, the complexity in creating the required control logic plays a dominant role in Flash memory device design. [11] When we zoom in on a sector we get the simplified layout depicted in Figure 3.10. We will discuss Flash cell designs and electrical cell models in the next chapter.

Electrical Flash Cell Modeling 4

Now that we have discussed the Flash cell, array architecture and functional modeling we will look at Flash cell models that have been published. By analyzing these cell models we will get a better understanding of creating an electrical Flash cell model based upon the industry standard Flash cell. This will enable us to create net list that we can simulate.

In the Section 4.1 we will discuss an overview of electrical cell modeling. In Section 4.2 we will discuss the published models and in Section 4.3 we will propose our own electrical model based upon the industry standard Flash cell.

4.1 Overview Flash Cell Modeling

Modeling semiconductor devices can be done by targeting fundamental physics or behavior. When creating an electrical model, two kinds of simulation approaches can be distinguished, namely physics driven and *CM* (Compact Model) driven. [5] Electrical models can be simulated with CAD tools by approaching the geometrical layer beneath or the behavior layers above. The former is a technology perspective and is concerned with accuracy of physical description. These models tend to be heavy in computation. In this thesis we are interested in finding an electrical model for a Flash device based on its behavior and we therefore we do not have to rely on underlying physical models. CM will therefore be sufficient for our discussion. *SPICE* (Simulation Program with Integrated Circuit Emphasis) simulators offer both physics driven and CM simulation possibilities. We will use HSPICE to create an industry standard Flash cell based CM.

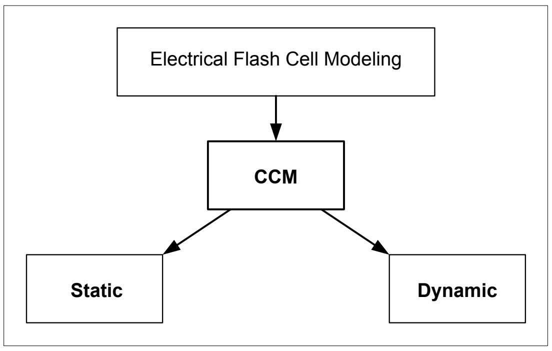


Figure 4.1: types of electrical Flash cell modeling

Figure 4.1 gives an overview of different types of electrical Flash cell models. We have discussed the CCM in Section 2.1.1 which ensures the change in threshold voltage in a Flash cell. Basically, the FG character is being described in CCM and we take into account the capacitive character of the Flash cell. On the left we have a "Static" type and on the right "Dynamic". The static type model has two different models, one for reading and one for writing. Writing is done by using a separate source to change the threshold level of the cell. We can then read the cell. The dynamic type model is able to do the same with one model. This model has a component that changes the threshold level of the cell and can then be read directly, i.e. we do not have to program a source to change the threshold level of the cell because a special writing component acts as the FG. We have discussed the writing equations that this component must use in Chapter 2.

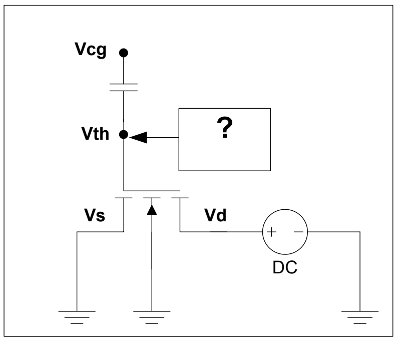


Figure 4.2: FG component(s) to simulate writing mechanism

If we look at an NMOS component in SPICE, we need to determine what component(s) to use to create the CCM effect. SPICE does not have a specific component that models a specific Flash writing mechanism. Our discussion of the published models will give us a better idea of what is common practice in Flash cell modeling. Figure 4.2 depicts the component to act out the writing mechanism that changes the threshold voltage of the cell. This can be accomplished by forcing the Vth level up or down with a voltage source (i.e. Passive) or by a component that changes the Vth level according to the writing levels on the nodes Vd, Vcg and Vs (i.e. Dynamic).

4.2 Published Flash Cell Models

In this section we will discuss 3 papers and 2 thesis' and a Flash book that describe electrical Flash cell models. They are ranged from the year 2000 to 2007 and will give us a balanced view of electrical Flash cell modeling practice in the academic field. After our discussion we will categorize them in types and in the next section we will propose an electrical model based upon the industry standard Flash cell.

4.2.1 Horng et. al.

Horng et. al. have published a paper called "A Realistic Fault Model for Flash Memories". [7] Their discussion dives into 6 classified types of fault behavior and simulation of these for the NAND type cell. We will however only focus on the electric model and this is depicted in Figure 4.3.

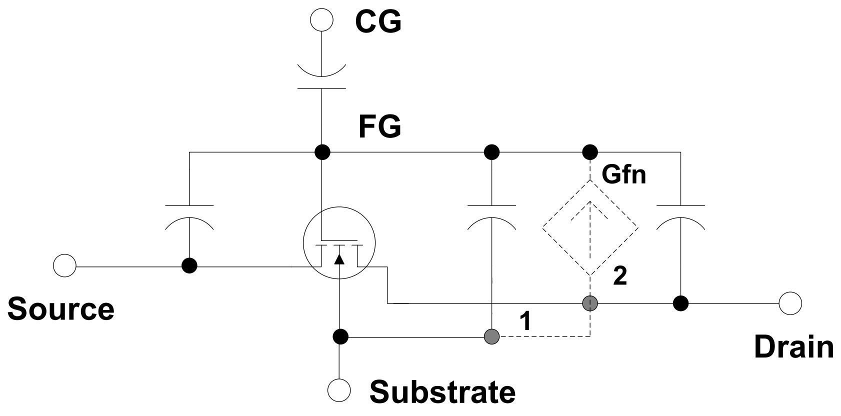


Figure 4.3: electrical Flash cell by Horng et. al.

We see that the CCM has been implemented not only by the NMOS and C_{cg}, but also by the additional capacitors that are connected to the FG. These capacitors create a voltage from Gfn and change the threshold of the NMOS cell. Also, we see that Gfn is either connected to the channel (number 2) and the bulk (number 1). This is because they propose 2 different models, the first having Gfn connected to the bulk and the second to the channel due to all the different fault types that are represented in their discussion. Gfn in this case represents the FN tunneling mechanism for programming and erasing the NAND cell. This is a situation where the Gfn component has to be triggered by a pulse to write the cell. [7] In this case, we do not have a cell that can be programmed, read, erased and read by only changing the voltages on CG, Source and Drain. It is not clear from this how Gfn effect will change the threshold level. We will therefore label this cell as a "Static" model.

4.2.2 Mohammad

Mohammad has written a Thesis called "Flash Memory Disturb Faults: Modeling, Simulation, and Test". [20] His discussion is detailed on disturb faults and his electric model is depicted in Figure 4.4.

We see that the CCM has been implemented by an NMOS and C_{cg} and that the writing mechanisms are depicted by 3 current sources. The electrical cell depicted is suited for industry standard cell simulation due to FN Tunneling and CHEI models that are applied. There is a detailed discussion on the equations applied. However, one can

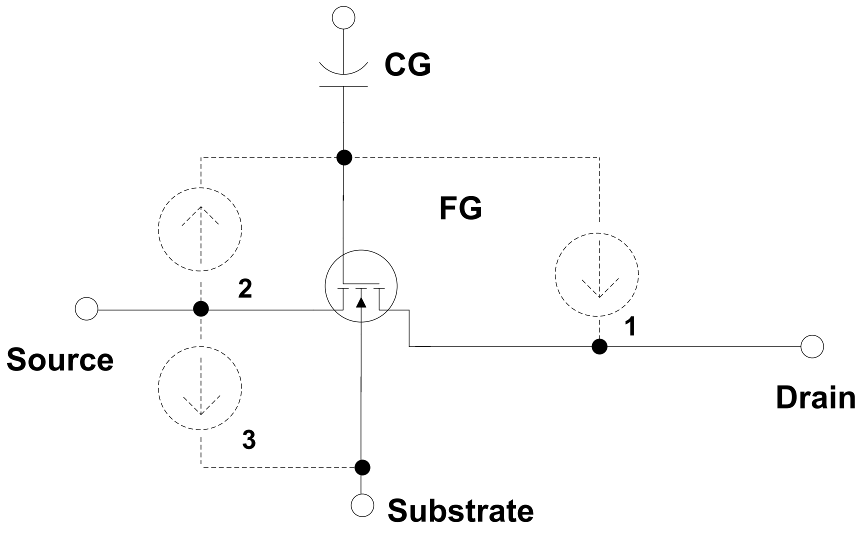


Figure 4.4: electrical Flash cell by Mohammad

only either write or read when simulating this cell, not both in the same model. Pulses are given to the current sources with the calculated models of the writing mechanism applied. In the case depicted in Figure 4.4, after we write, the threshold voltage change is not retained. We will therefore also label this cell as a "Static" model.

4.2.3 Larcher et. al.

Larcher et. al. have written a book written a book called "Floating Gate Devices: Operation and Compact Modeling". [11] With detailed discussions on writing mechanisms and CM for Flash devices, their electrical model is depicted in Figure 4.5.

Here we see similarities with the model proposed by Mohammad. The biggest difference however is that Larcher et. al. have chosen to use a voltage source (V1) to change the threshold voltage. The current sources simulate the writing mechanisms. It is a totally different approach that gives a more dynamic model but still is dependent upon timed pulses to make sure that all nodes act according to a written situation. This model however leads to better accuracy and less computation [11] and which is very easy to simulate in SPICE. We will however label this model as "Static" due to the 'programming' one has to do on the gate with a voltage source in order to see the right cell response (i.e. keep threshold voltage either low or high during reading).

4.2.4 Kang et. al.

Kang et. al. have written a paper called "A Simple Flash Memory Cell Model for Transient Circuit Simulation". [9] Their electrical model is depicted in Figure 4.6.

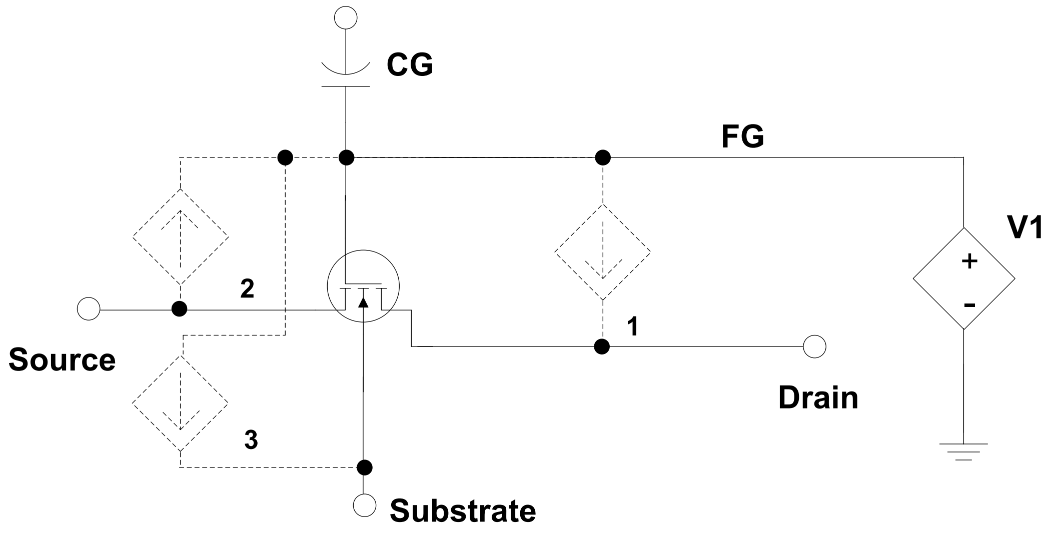


Figure 4.5: electrical Flash cell by Larcher et. al.

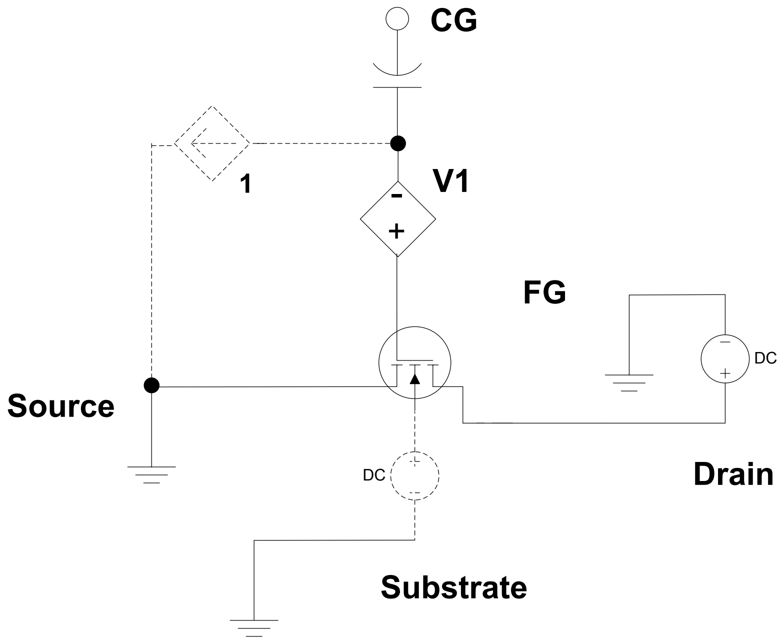


Figure 4.6: electrical Flash cell by Kang et. al.

Kang et. al. has almost the same approach as Larcher et. al. regarding the threshold voltage using voltage source V1. We see that the bulk and drain are connected to voltage sources to ensure the effects of specific writing mechanisms. Also, the current source is added to ensure the writing mechanism current. The same as with Larcher et. al. applies to this model however. We will therefore also label this as a "Static" model.

4.2.5 Ginez, Mauroux et. al.

Mauroux et. al. have written a paper based upon a model that was described in a Thesis of Ginez. Both target ATMEL embedded Flash memory with a FLOTOX (Floating Gate Tunnel Oxide) cell which has a different structure than the industry standard cell. [13] [15] Because of this structure it uses FN Tunneling for both programming and erasing, which (due to lower power consumption than an injection mechanism) is preferable for embedded memory. The electrical model is depicted in Figure 4.7.

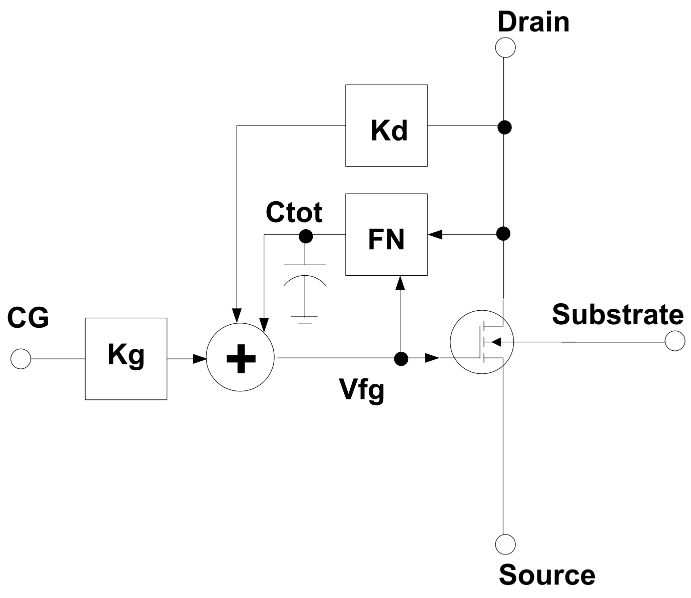


Figure 4.7: electrical Flash cell by Mauroux et. al. and Ginez

As we can see the structure of this model introduces new components and it actually introduces a feedback loop from the gate back to the FN block. The output of this FN block represents the FN Tunneling current. The feedback loop generates a dynamic situation that works due to the difference between the gate (FG) and the channel. The FN block is a behavioral component that is described by Equation 2.14. [15] We can implement behavioral blocks in SPICE by using Verilog A (hardware description language) or as 'VALUE' type equations (behavioral modeling in SPICE language) based upon the different node voltages and FN constants. We can either make blocks that output voltages or currents. Kg and Kd represent coupling factors for the Drain and CG voltage, and the circle in the middle is a simple voltage summer. All these components are easy to implement in SPICE. The FN current being fed to Ctot generates a voltage and based upon the difference between Vfg and the voltage in the channel, the threshold voltage of the cell is able to change. This is a "Dynamic" situation because we do not have to program a specific node to act as if written. The feedback loop simply changes the threshold voltage at the FG due to the generated FN block voltage.

4.2.6 Published Overview

All the Flash cells that we have discussed are "Static" in nature except for the model proposed by Ginez. This is depicted in Figure 4.8. Apart from the specific memory technology that is targeted by Ginez a dynamic type of electrical Flash cell has been proposed. Our aim will be to create a dynamic type of industry standard Flash cell based upon our discussion.

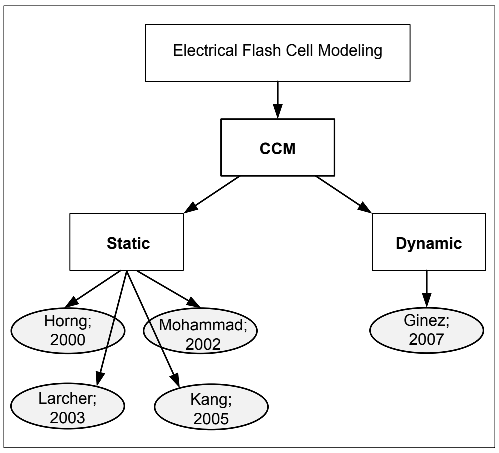


Figure 4.8: published Flash cell overview

4.3 Industry Standard Flash Cell

By combining the model made by Ginez with a block that represents CHEI we get an even more general circuit. This will be suitable for an industry standard cell model. This model is depicted in Figure 4.9. This structure is able to represent CHEI and FN writing mechanisms instead of only FN presented by Ginez, Mauroux et. al. Note that due to the 2 writing mechanism blocks, Ctot has been split up into C1 and C2. This is due to the voltage that needs to be generated by the writing mechanism current at both these nodes.

We will have to create a working SPICE simulation to verify the model. From the discussion in the literature it is not yet clear how the feedback loop works in SPICE simulation. Now we need to find out how to implement this in SPICE simulation by creating a net list. We will discuss this in the next chapter.

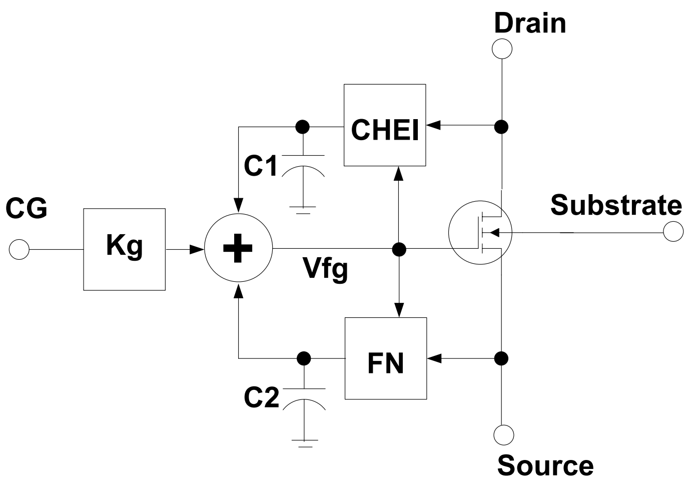


Figure 4.9: industry standard electrical Flash cell

Modeling Flash Memory 5

Up until now we have only discussed concepts of models that represent the Flash cell and the functional blocks of a Flash memory device. We have not yet seen an electrical equivalent for simulation. Based upon previous discussions we will create these models in SPICE. This will enable us to simulate the industry standard Flash cell with FN and CHEI writing mechanism. Also, we will discuss the NOR and NAND array and look at their simulations in HSPICE.

In Section 5.1 of we will discuss the different Flash blocks and their electrical equivalents in SPICE, this prepares us in creating a complete BFFM with NOR and NAND array architecture. Section 5.2 starts with a 1 bit *ISFC* (Industry Standard Flash Cell) equivalent for simulation. In Section 5.3 we will look at a 2x2 bit NOR array model and we will discuss the 2x1 bit NAND array model in Section 5.4.

5.1 Flash Memory Periphery Blocks

Using the BFFM layout we discussed in Chapter 3 and the discussion we had on Flash cell modeling in the previous chapter, we can start creating a net list in SPICE. The Flash cell net list is hardly discussed in literature. We have a description of a cell by Mohammad and a detailed discussion by Larcher et. al., however we have labeled these models "Static" due to the programming of sources to change the threshold level of the cell. So we will create our own dynamic model based upon the description of Ginez and Mauroux. Let us look at the three electrical parts that make up our BFFM layout.

5.1.1 Decoder

For the decoder we need to create a buffer or voltage source that targets the writing mechanism and selects a specific cell and also sets a voltage for reading. We can use straightforward components to enable this. Figure 5.1 depicts a buffer with 2 inverters in cascade. We will use a voltage source if the voltage varies much e.g. high for programming and lower for selecting the cell, as is the case for the control gate voltage.

5.1.2 Flash Cell

The Flash cell we are interested in needs to be suitable for both NOR and NAND writing mechanisms. We will therefore create a compact model for the ISFC suitable for both array types. Figure 5.2 depicts both ISFC types. The key feature of our Flash cell is changing its threshold voltage due to programming and erasing. Getting this to work with components that insert currents (due to the writing mechanism) will need a

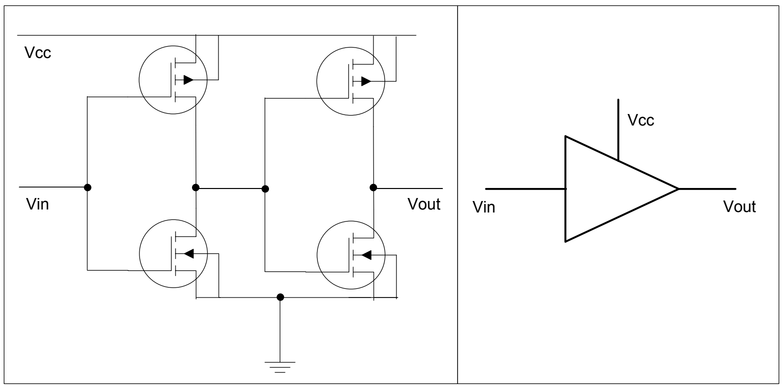


Figure 5.1: buffer - 2 cascaded inverters

capacitor as depicted in Figure 5.2 to create a voltage. This voltage is either positive or negative and therefore changes the threshold voltage at the FG. However, the equations we have seen in Chapter 2 suggest an electric field, E_{ox}. In the case of ISFC1 this field is created by programming with CHEI (P:CHEI) having a field between drain and FG, and by erasing with FN (E:FN) having a field between source and FG. With ISFC2 both programming and erasing is done by FN and the change in threshold is due to field between drain and FG.

When first trying to recreate this model it not yet apparent that the field between the channel and the FG holds the key in this feedback loop. Equation 2.16 gives the field description where V_{app} represents the voltage at the FG and V_{fb} the voltage at either drain or source depending on the targeted writing mechanism. The feedback loop then feeds this change of voltage between the nodes towards the FG, thus changing the threshold voltage. In case of FN, the voltage at the FG drops due to the feedback thus programming the cell by increasing the threshold voltage. However in ISCF2 Vbitline uses FN to erase the cell by decreasing the threshold voltage and setting a more positive voltage at the FG. Note that a high threshold voltage is due to a negative offset at FG and a low threshold voltage is due to a high offset voltage. The offset voltage is set by the mechanisms used. With CHEI the voltage at the source, Vsource, is high and therefore sets a low threshold voltage for the cell. We have the equations for both FN and CHEI described in Chapter 2 and Figure 5.2 depicts where they play a role. Kg is a coupling factor that we can view as a factor of the CG voltage. Note that we have discarded the coupling factor of the drain (Kg) because it is a small factor that is compensated by the FN component. We can simply add this block by adding it to the net list as a dependent voltage source that has represents a value of the voltage at the channel multiplied by a specific coupling factor. We have also added a bitline capacitance to the drain of our cell. This aids in smoothing the voltage curves of the channel and also reflects the physical reality as discussed in Flash literature. [4] In the Appendices the interested reader can look at the net lists discussed in this chapter, as well as all the dimensions and parameters of this net list.

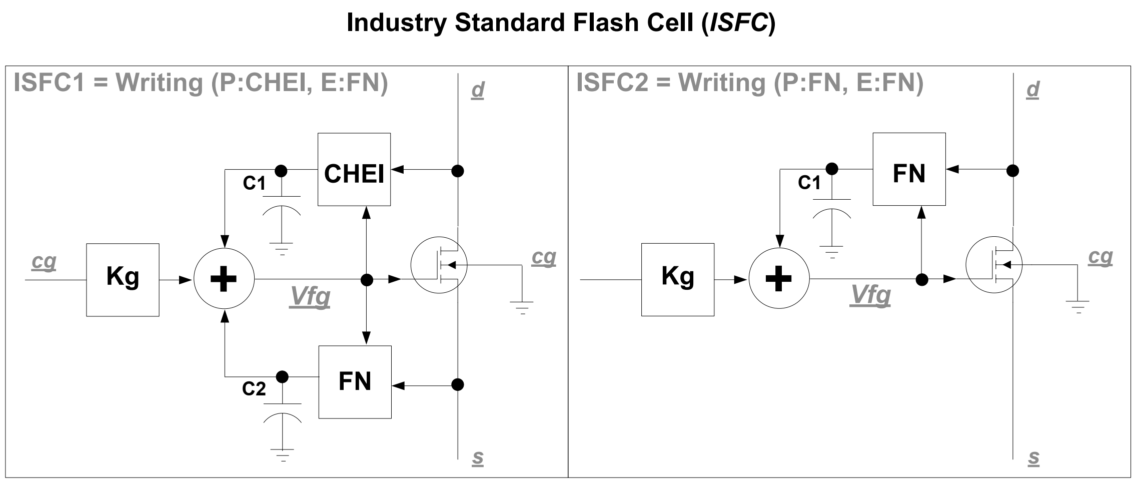

Industry Standard Flash Cell (ISFC)
ISFC1 = Writing (P:CHEI, E:FN)
ISFC2 = Writing (P:FN, E:FN)
d
s
cg
Kg
+
Vfg
C1
C2
CHEI
FN

Figure 5.2: ISFC types

5.1.2.1 Comparison to Ginez, Mauroux et. al. Model

If we compare our ISFC model to the one proposed by 'Ginez, Mauroux et. al.', we see that the total capacitance has changed into C1 and C2 for the ISFC1 type. This results in slightly increased simulation time from 0.21 seconds with the 'Ginez, Mauroux et. al.' model, to 0.33 seconds with the ISFC1 type. This is due to the added capacitor and CHEI block. The ISFC2 type however does not have an increased simulation time. We can therefore conclude that in our ISFC model with a relatively small number of cells, simulation time has hardly changed compared to the 'Ginez, Mauroux et. al.' model. In this Thesis the number of cells during simulation will not exceed 2x2.

5.1.3 Sense Amplifier

Reading the logic level of a cell is the last stage of our BFFM. In Flash literature we mostly read about solutions with comparators, [4] has very detailed descriptions on different kinds of architectures with these blocks. We will create a situation where we can clearly distinguish between logic levels of a written cell. Because we are reading at the drain of a bitline, we must use a component that does not consume too much power. An active reading component will therefore be needed. By using a latch in this comparator we will try to set a clear difference between the logic stages for the output. We can use a simple latched comparator design to achieve this result.[19]

Figure 5.3 depicts the comparator we will use in our net list. The latch will be triggered right before and right after reading the logic level. Vcc is the level of our output and we will only use Vout+. Vin- is the reference voltage also described in Figure 2.3. The sensing voltage here comes from our bitline component and results in either the cell conducting towards ground (logic level 0) or the cell being open (logic level 1). The read voltage between these two levels is used as our reference voltage. Let us look at complete memory model and simulations.

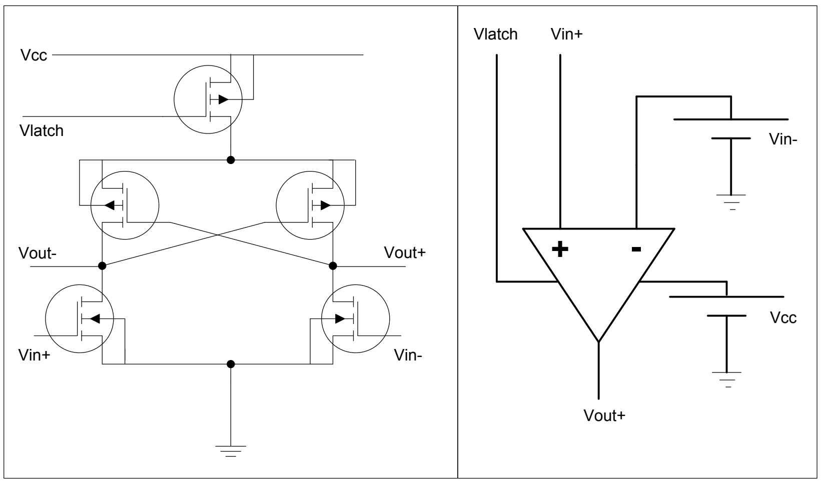


Figure 5.3: sense amplifier - latched comparator

5.2 Flash Memory Model: 1 Bit ISFC

Combining all the previous blocks to create an electrical equivalent of the BFFM means setting up the decoder, cell and sense amplifier with two different cell types. One suitable for writing according to NOR array and another according to NAND array architecture. The NOR array architecture requires the cell to be programmed by CHEI and erased by FN. The NAND however uses FN for both. Figure 5.4 depicts the NOR array cell type of writing. We will call it type ISFC1 and the NAND array type will be termed as ISFC2. The latter is depicted in Figure 5.5.

Figure 5.4 depicts the Vcg (control gate) voltage source that must vary in voltage. On one hand it needs to be set high in order to trigger the CHEI mechanism for programming and on the other it needs to be set to a lower voltage in order to supply the sensing voltage for reading. When the erasing mechanism is triggered it is set to ground. Vsource is responsible for the FN mechanism and is set to ground during all other events. Vbitline ensures the reading voltage through the channel. It also plays in role in programming the cell by simultaneously with Vcg setting a voltage similar to the read voltage through the channel. This ISFC layout is based upon an Etox cell, also discussed in Chapter 2. Figure 5.6 gives an overview of the different states and the voltages on the drain, source and control gate must be in the net list.

Figure 5.5 depicts a slightly different situation. Vcg is responsible for erasing and reading the cell instead of programming. Vbitline is the source that erases the cell. Also, it sets the reading voltage throughout the channel. There is no Vsource used in this array type and the cell is directly connected to ground. This means that cells in serial have to be able to set in pass though mode. This can be done by setting the cell

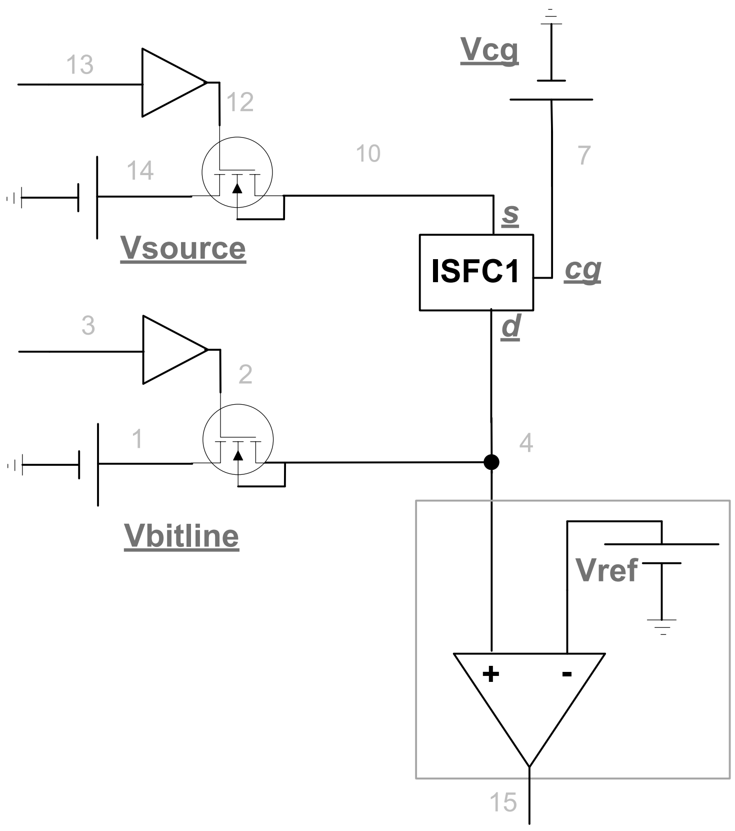


Figure 5.4: NOR array type BFFM - 1 cell ISFC1

in saturation mode (open). Vcg is then set to a voltage that ensures an open channel even when the threshold voltage has been increased by programming. Figure 5.7 gives an overview of the different NAND states. We see the extra state depicted to ensure the serial setup of a NAND array bitline. Note that this cell has been coined by Ginez as Flotox in the previous chapter, an embedded memory variant. Our ISFC2 type is an embedded type of Flash cell with complete FN writing to ensure low power consumption.

We have used voltage source with variable voltage levels for the Vcg sources due to simplicity in running simulations. We have used the buffers where the sources were set to one voltage level. Both represent a decoder type of voltage with the purpose of selecting the needed nodes. The comparator is the same component as described before with a Vcc of 2V. This will ensure a logical 0 and logical 1 (approaching the 2V). Let us look at the simulations the net lists that are based upon the discussed figures.

5.2.1 Simulation Results

For all the operations in our 1 bit memory we can look at Table 5.1. This gives a summary of the sources and nodes that are of interest here. We will use our net list to simulate Program, Erase and Read. Figure 5.8 depicts the control gate, bitline source

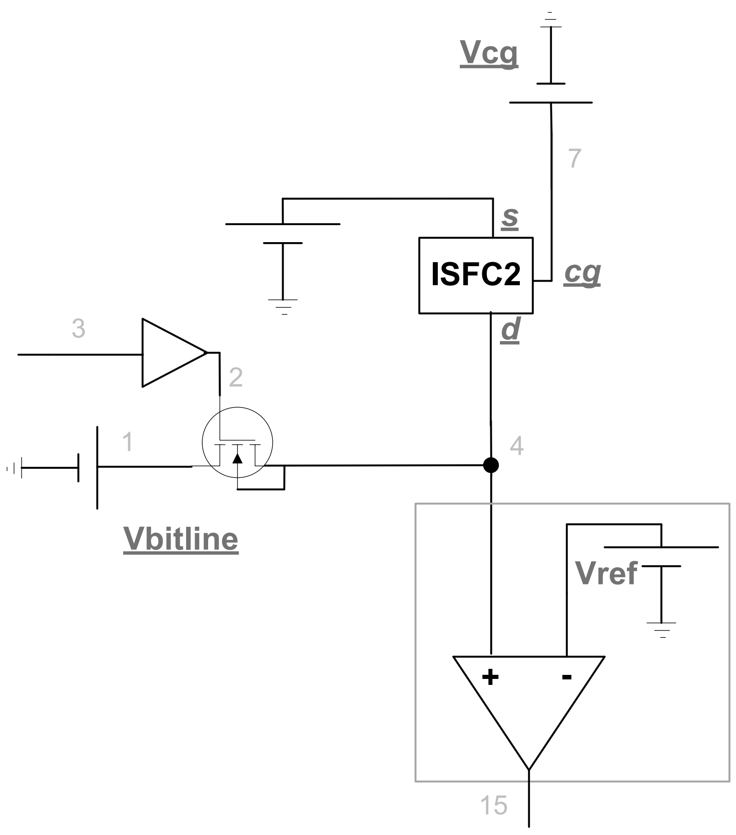


Figure 5.5: NAND array type BFFM - 1 cell ISFC2

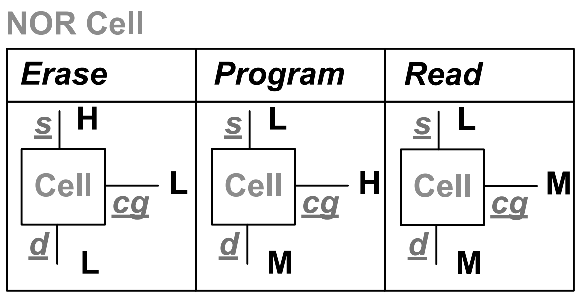


Figure 5.6: NOR cell states

and latch voltage of our net list of the 1 bit ISFC. They reflect the table values in all modes of operation. After each read we see that the Vlatchcomp is high to reset the comparator value for the next logic level reading. We have a Program, Read, Erase, Read, Program and Read sequence to clearly show the output of this cell and the voltage level at FG due to the writing mechanisms.

Figure 5.9 depicts what is happening on the FG due to the FN and CHEI component in the net list. The threshold voltage is high after programming the cell which results in a

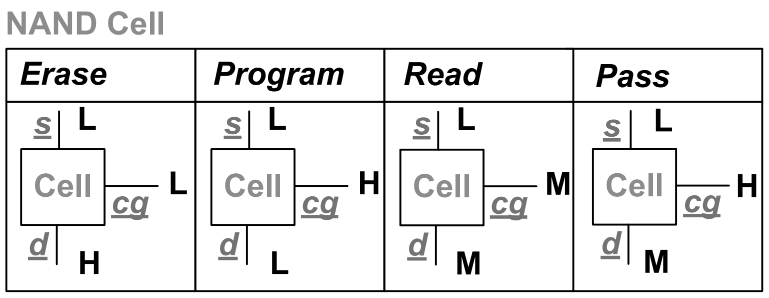


Figure 5.7: NAND cell states

Table 5.1: 1 bit ISFC table

	Vcg	Vbl	Vsrc	Vout	Vref
Program	3.5	1	0	-	-
Erase	0	0	8	-	-
Read	1	1	0	0, 1.6	0.75

lower voltage at FG, Vfloatinggate. Th_h depicts the cell's high threshold level, resulting in a lower voltage at the FG. Th_l depicts a low cell threshold due to erasing. The channel depicts what the logic level is by showing a cell that has logic level 1 after programming and 0 after erasing. After connecting the drain channel node to the comparator we set Vref (Vin-) to 0.75V. This will ensure that only everything above this value will be set to Vcc, which is logic level 1. This is clearly depicted as Vout in the lower part of the figure.

Simulating ISFC1 needs an extra component compared to ISFC2. This extra component is CHEI. In our compact model we have simplified this equation to reduce complexity in modeling. Because we are not interested in accurate physical behavior this simplified version will be sufficient. The interested reader can simulate the model of this discussion using the net list in Appendix A. Let us look at a more complex models next.

5.3 Flash Memory Model: 2x2 Bit NOR

Simulating only 1 cell is limiting when it comes to the electrical behavior. This is due to the array structure we are then dealing with. Only when we discuss an array of cells will we get more acquaint with these structure. If we regard a 2x2 NOR array as complete block, we should have the possibility to program all cells. Figure 2.12 depicts this structure and we see that the control gate (or word lines) are connected to each other. This however makes it difficult to program only 1 cell of the 2 bit word. We will therefore chose to have separate word lines for programming. The 2 bit word is however read simultaneously. Erasing all cells should also be possible simultaneously. All these matters should be taken into account in the net list. Figure 5.10 depicts this net list. Cells A1 and A2 are a 2 bit word. This also applies to B1 and B2. Writing

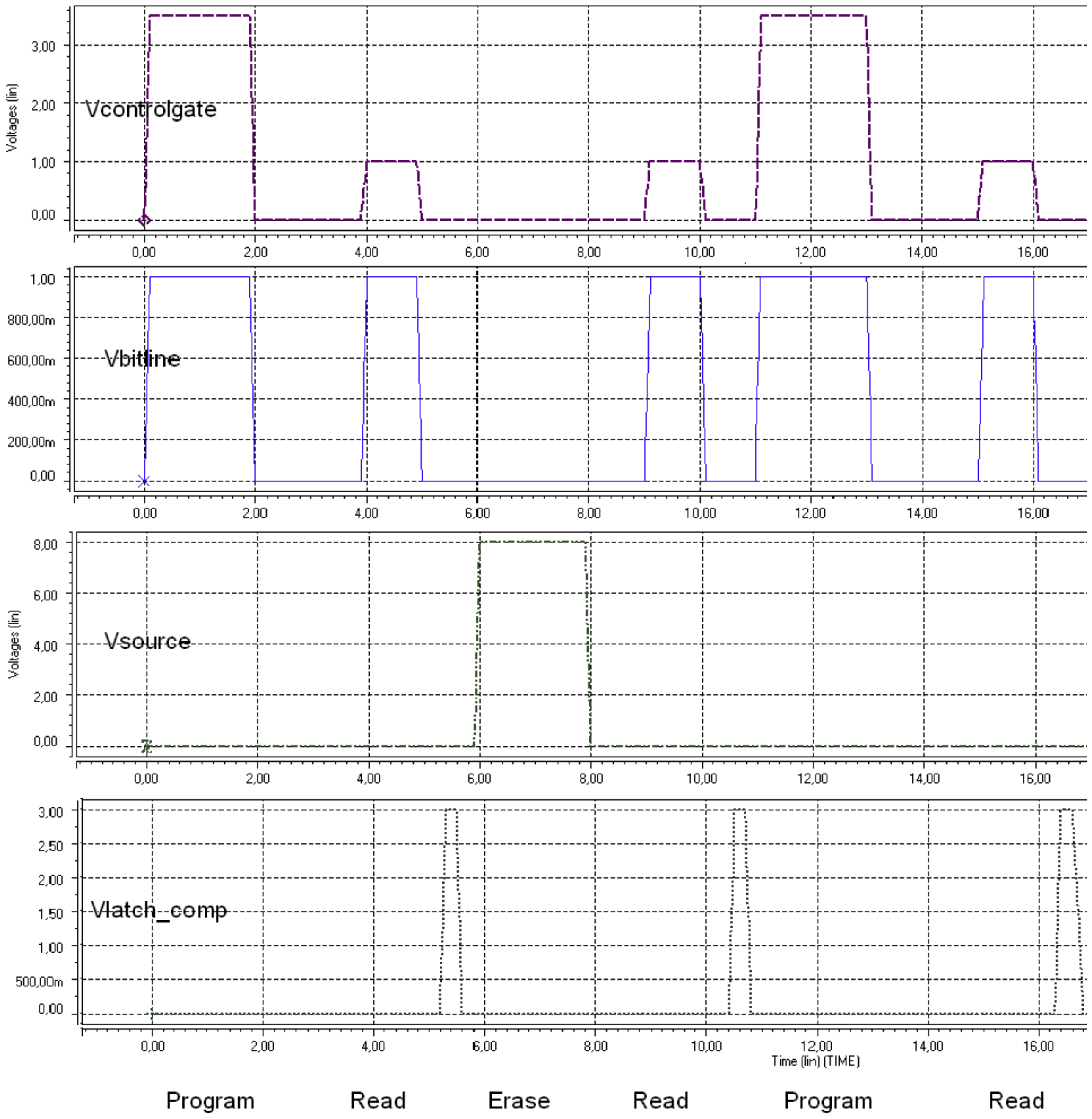


Figure 5.8: 1 bit ISFC: program, erase and read - 1/2

methods in Flash memories are described in the flowchart depicted in Figure 2.10. For our simulations we will not create complex writing mechanism logic in order to check the state of cell and we will therefore create a simple schema for our simulations. We can both erase and program all cells simultaneously so we might as well start by erasing all cells and then program the specified ones. We can afterwards read their logic level.

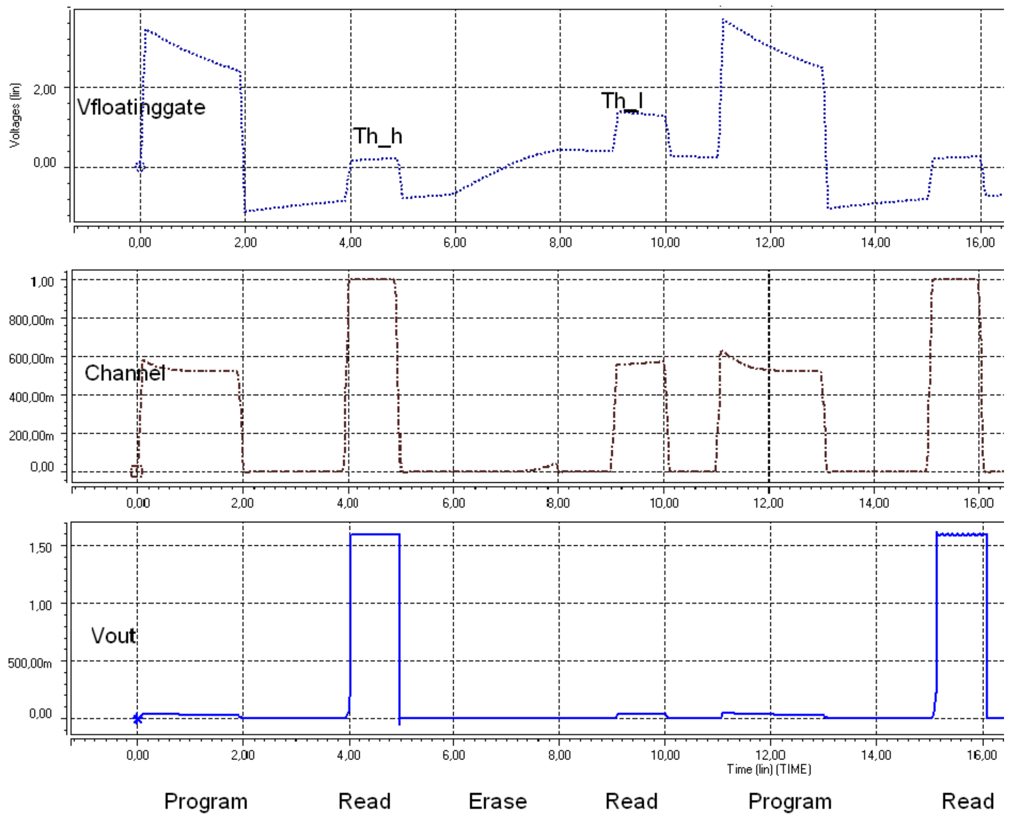


Figure 5.9: 1 bit ISFC: program, erase and read - 2/2

We agreed to first erase the cells, we can keep the number of Vsrc sources to a minimum in accordance with Figure 2.12. In the NOR figure we see that we have 1 Vsrc source for a pair of NOR cells. Other than having more sources, 2 bitlines and comparators, the structure is similar to the 1 bit simulation discussed in the previous section. The 2x2 NOR array gives us insight in how an nxn NOR net list would function. Because we have 2 comparators we will first read word A and then read word B. Writing a block therefore results in n reads for our nxn model. We will look at these simulations next.

5.3.1 Simulation Results

In our 2x2 NOR net list we use the values depicted in Table 5.2. We have used different values for programming and erasing and changed the factors of the nodes in the FN and CHEI blocks. These changes have been made during tweaking of the cell's logic level and is only necessary to give a solid outcome. One may tweak the compact model according to the number of components added and the results one needs. Appendix B depicts this net list for our simulation results. We have an Erase, Program(A1, B2), Read(word A), Read(word B), Erase, Program(A2, B1), Read(word A) and then Read(word B). This will give us a good overview of the NOR array.

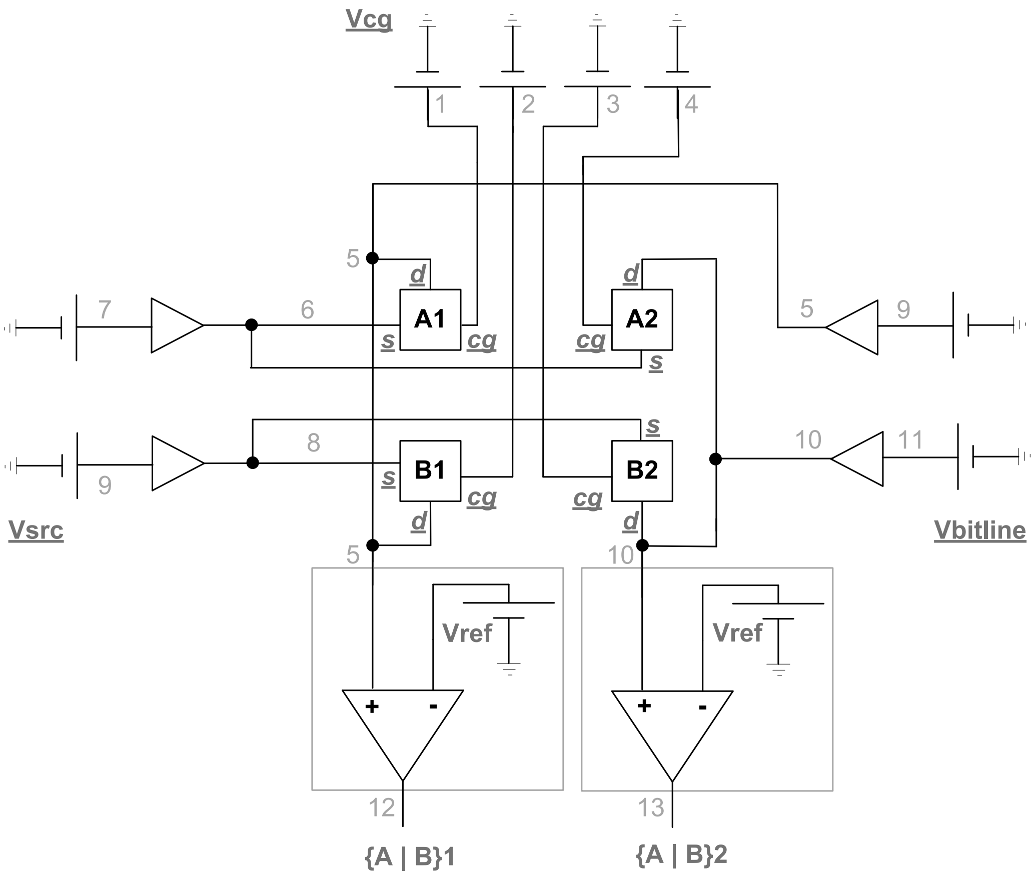


Figure 5.10: 2x2 NOR array

Table 5.2: source values of the NOR array

	Vcg	Vbl	Vsrc	Vout	Vref
Program	3	3	0	-	-
Erase	0	0	3	-	-
Read	1	1	0	{0, 1.8}	0.5

Figure 5.11 depicts the values of the sources that are needed to Erase, Program and Read the cells in 2 iterations. We also see that the program levels of Vcg2 and Vcg3 have an increasing slop of 0.5V during their pulse. This is because these cells (B1 and B2) are read in the second iteration. We see that the value of FG is affected by change on the channel in the meantime and we therefore need to slightly increase the voltage to ensure the right level. The FG levels are depicted in Figure 5.12. Note that is the number of words increases, the Vcg should as well to ensure the correct level at read time. Also, the bitline values for both channels are exactly the same. The source values are similar to each other too.

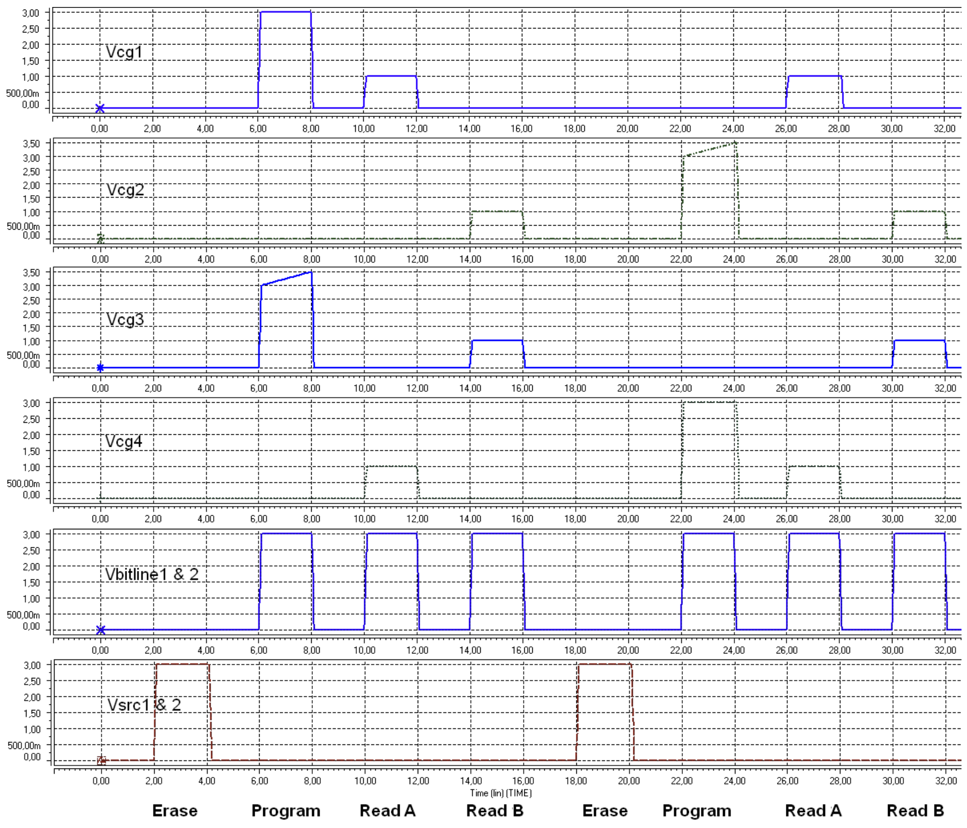


Figure 5.11: 2x2 NOR: erase, program and read - 1/2

In Figure 5.12 we see the second part of the simulation. The levels of VfgA1, VfgA2, VfgB1 and VfgB2 reflect how the writing mechanisms influence the cells. The threshold levels are clearly changed here. We see for example at cell A1 that VfgA1 goes up in voltage after the Erase between 2 and 4 seconds, and then it goes down after programming between 6 and 8 seconds. Reading cell A1 after the program results in logical 1 in the Vout n1 graph between 10 and 12 seconds. The same principle applies to all the other cells. Channel A and B show the raw footage of the cells and we clearly see the comparator in the last 2 Vout graphs filtering out anything below the 0.5V (Vref) on the channels. At Vout n1 the first read iteration gives a binary 10 and at the second it gives a 01. At Vout n2 we get 01 and then 10. This gives us the result in Table 5.4. Let us look at how a 2x1 NAND net list functions next.

Table 5.3: 2x2 NAND simulation results

	Bit 0	Bit 1
Word A	1	0
Word B	0	1

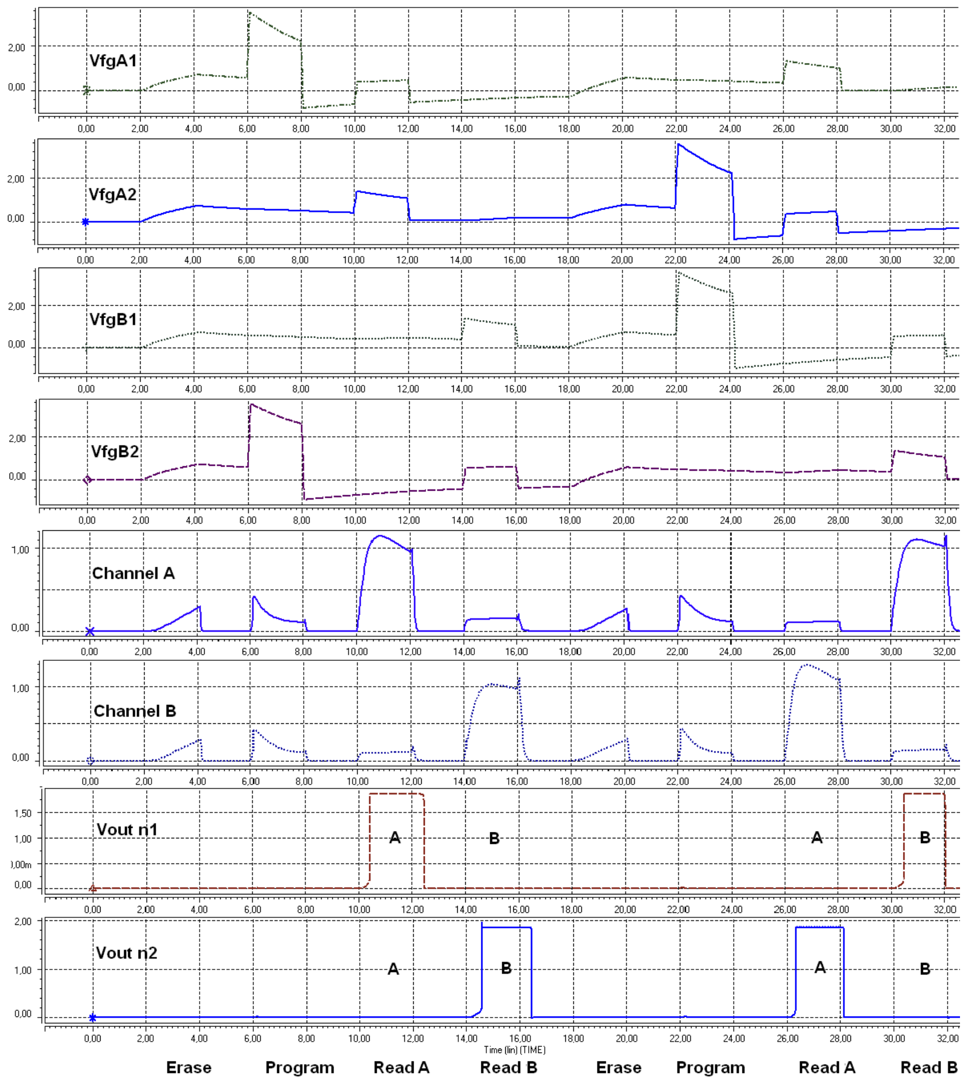


Figure 5.12: 2x2 NOR: erase, program and read - 2/2

5.4 Flash Memory Model: 2x1 Bit NAND

For our next discussion we will look at a model that simulates 2 words, 1-bit NAND array. We have already discussed a simulation with 2 channels (2 bit NOR) and therefore we will now focus on the NAND pass-through functionality, needing at least 2 words for this. Figure 5.13 the NAND net list. We see that we do not have Vsrc for erasing the cells and that the cells are connected in serial, i.e. the drain of A1 is connected to the source of B1. Reading A1 will therefore require B1 to be in pass-through and reading B1 requires A1 to be in pass-through, as if B1 was directly connected to ground. The same needs to happen for any n-word length. The higher the number of words, the more we deal with the capacitive nature of the cells and this will need more tweaking then in the net list. In simulating these fairly simple circuits we already deal with a lot of tweaking. One can imagine that this effort and complexity increases dramatically when dealing with real cells in manufacturing.

For the NOR net list we have chosen to first erase all cells and then programs the ones specified. However in our NAND design depicted in Figure 5.13 we immediately see that erasing the cells must be done in parallel, because we need to set at least one cell in pass-through to target the one behind it in the chain. We then have to look at another mechanism that can be done in parallel to invert our approach. Fortunately this is possible by the separate control gate lines (Vcg). We can program all cells simultaneously. We will then invert our approach and first program all cells and then erase the ones specified. By creating a net list we see that we can more easily determine a working writing approach for different array architectures. For simplicity we have used only sources in our net list, we have already seen a more complete decoder design with buffers in the NOR model. For the results in our NAND simulation this will not be necessary. The interested reader can find the net list of this design in Appendix C. Let us look at the NAND simulations next.

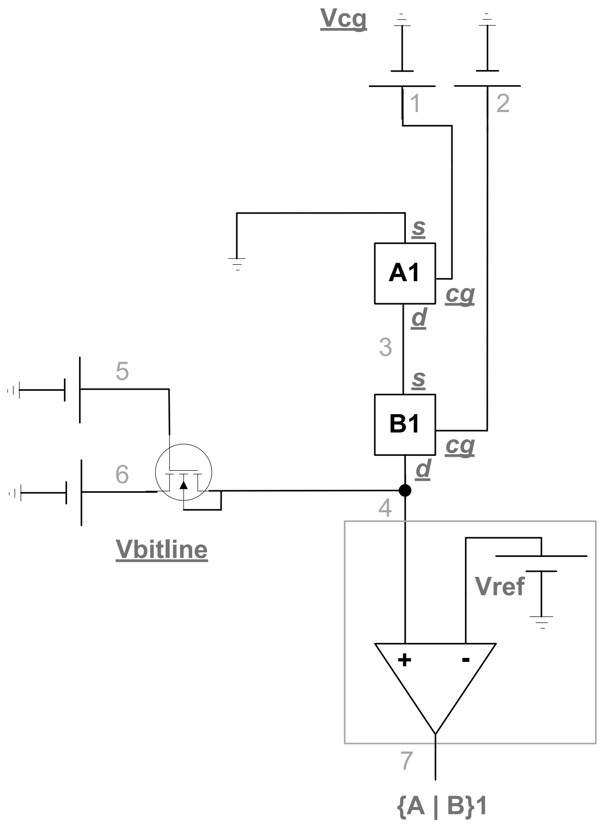


Figure 5.13: 2x1 NAND array

5.4.1 Simulation Results

Table 5.4 depicts the values used in the NAND net list. We see a situation where the bitline voltage would be 3 when programming and when reading. This is because we need to set Vbitline to 3 in order to put all other cells than the one targeted in pass-through mode. When erasing we are limited to the number of words in a bitline because the Vcg of the targeted cell must be 0. Putting a cell in pass-through however requires Vcg to be 3V so simultaneously erasing all cells is not possible. Having n words that have cells will thus result in n erases via this design. We also see that reading a cell requires Vcg to be 1V (T:1) and the other cells in pass-through to 3V (P:3). So the pass-through has a Vbitline of 3V for reading and simultaneously setting cells in pass-through. In operation mode this array type is more complex than the NOR. Appendix C depicts the NAND net list. In this simulation we will first Program, Erase(A), Read(A) and then Read(B).

Table 5.4: source values of the NAND array

	Vcg	Vbl	Vout	Vref
Program	3	3	-	-
Erase	0	5	-	-
Read	T:1, P:3	2	{0, 1.6}	0.8

Figure 5.14 depicts the values of Vcg and bitline. After programming both 1-bit words we erase A1 and put B1 in pass-through. We can see this in the Vcg levels. B1 is in pass-through by setting Vcg2 to 3V. The Vbitline of 5V will then effect cell A1 due to Vcg1 being 0V, the electrical field between the channel and the FG will result in lower threshold voltage like we discussed before. As long as the channel and the FG do not differ much, the effect will hardly be visible due to the feedback loop (Vchannel and Vfg) we discussed. We then read both channels by reading A1 between 10 and 12 seconds and B1 between 14 and 16.

Figure 5.15 shows the results of the written NAND array. We see that the threshold voltage increase to the voltage drop on FG in VfgA1 and VfgB1. We then see VfgA1 increase due to erasing. VfgB1 is put in pass-through mode and is hardly affected. We then read A1 by putting B1 in pass-through mode again and then put A1 in pass-through to read B1. We can read a logical 1 and 0 accordingly. Our Vref in the comparator here is 0.8V.

Our discussion on the array types and simulating the net lists has given us insight in the electrical simulation of a BFFM Flash memory device. From here we can improve on the design and take even more different array types into account by using the same approach. We will conclude our discussion in the next chapter and formulate our conclusions and recommendations.

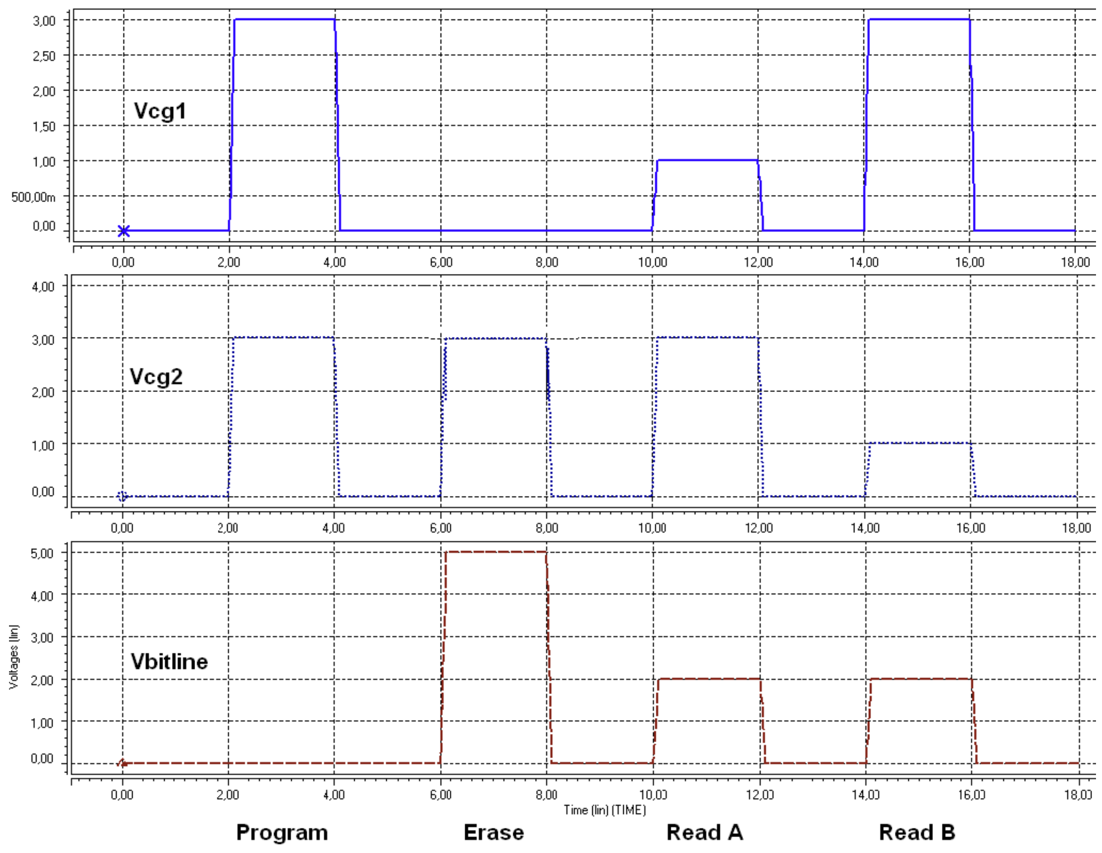


Figure 5.14: 1x2 NAND: erase, program and read - 1/2

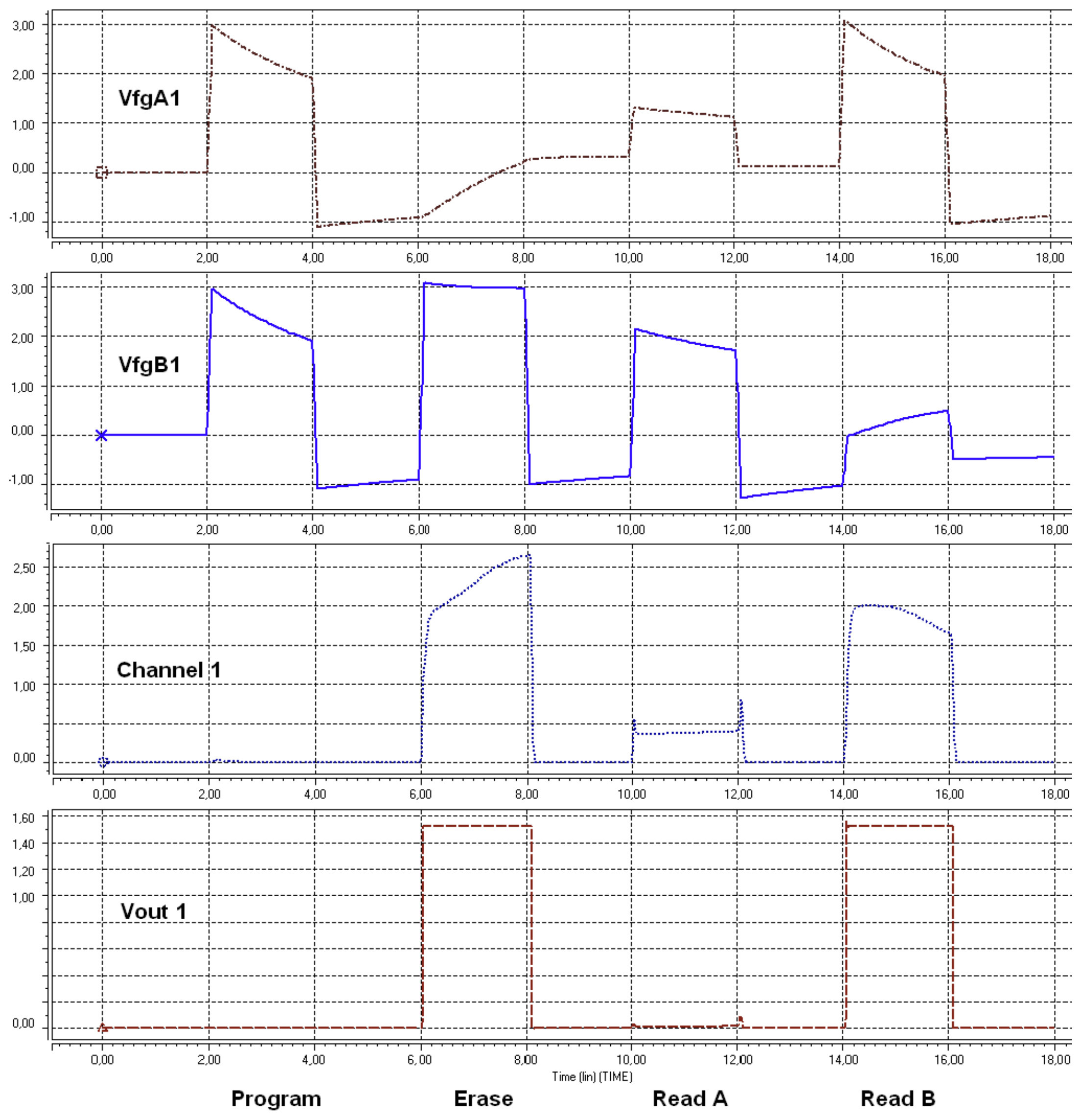


Figure 5.15: 1x2 NAND: erase, program and read - 2/2

Conclusions and Recommendations 6

In this chapter we will present a conclusion based upon this Thesis work. We will discuss the structure and the results from our discussion. Also, we will look at how our results can be improved for future work.

6.1 Conclusion

Our discussion on creating an electrical Flash memory device has lead us from generic concepts to a practical implementation of the Flash cell and complete memory device (based upon a specific array type e.g. NOR). We have seen the NMOS transistor being combined with a capacitive coupling model to present the Flash cell. It has no equivalent model in electrical simulation. Due to its complex capacitive nature it also hardly has any descriptions regarding electrical modeling in Flash literature. All explanations have been theoretical and suggestive regarding design, i.e. not discussing any details, just the results. We have found that the best descriptions have come from actual field data where the 'creature' has been studied in its habitat. [4] [11] The most recent descriptions have also come from researchers working with manufacturers and having access to actual memory data. [15] [13] [13] Their documents have been very useful in creating an equivalent electrical model. We hardly had any net list examples and made ours therefore by trial and error. This process has lead us to best practices regarding in Flash array type design. This has kept our model and net list simple and easy to reproduce.

We have created an electrical Flash memory device and equivalent net lists. It supports the FN and CHEI writing mechanisms. This device has been modeled based upon a functional memory design [16] in order to reduce complexity of our simulations. We have succeeded in created a model that is active in nature, meaning that we have created components that adapt to the electrical field of the nodes. This results in a logical 1 after being erased and a logical 0 after being programmed. We have based the active nature of our model on the description of Ginez [15]. Also, our model is suitable for NOR and NAND type array architectures. This gives us the flexibility to create and simulate nxn models of these different array types.

6.2 Recommendations

During our discussion of the net lists and different array types we have seen that reading the logic levels of the cells happens in iterations. We read words after one another meaning that a high number of iterations will result in channel with cells that are reacting on the difference between the channel voltage level and their FG. In reality, the FG and CHEI blocks are "switched off" if you will, after each writing mechanism. Now however, we see the FN and CHEI equations having effect on the FG voltage. this means that the cell that is read last, must be either programmed or erased with a higher value for it to retain the same level. This can result in a lot of net list tweaking when one targets a high number of words. An improvement would be to isolate the voltage on the FG somehow to have it shielded of from the electrical field when the cell is being read. This needs to be investigated further for it might pose a problem in creating net lists with a large number of cells.

As discussed in Section 5.1.2.1, one of the differences in our ISFC model is that in type ISFC1 we have split up the total capacitance in C1 and C2. Simulations have run successfully, however we have introduced two storage elements resulting in a second order circuit. We need to further investigate how to simplify type ISFC1 to a model that can be represented by a more simple first order circuit. This may prove to be advantageous in simulation time for a higher number of cells than discussed in this Thesis.

Creating the NOR and NAND array types has also lead to questions about the other types that are depicted in Figure 2.11, e.g. AND and DINOR. The different array types have their characteristic that may result in different writing schemes. The net lists of these models are not made yet and may be worth investigating for further electrical modeling purposes.

Net list: 1 bit ISFC A

```
** Flash Representative (decoder, flash cell(s), sense apmlifier)
** 1 bit memory model (writing schema(s): ISFC1 and ISFC2)

.MODEL NMS NMOS
.MODEL PMS PMOS

**————SOURCES (V)

Vbl 1 0 pwl(0 0 0.1 1 1.9 1 2 0 3.9 0 4 1 4.9 1 5 0 9 0 9.1 1 10 1 10.1 0
+11 0 11.1 1 13 1 13.1 0 15 0 15.1 1 16 1 16.1 0 20 0 20.1 1 21 1 21.1 0)
Vcg 7 0 0 pwl(0 0 0.1 3.5 1.9 3.5 2 0 3.9 0 4 1 4.9 1 5 0 9 0 9.1 1 10 1 10.1
+0 11 0 11.1 3.5 13 3.5 13.1 0 15 0 15.1 1 16 1 16.1 0 20 0 20.1 1 21 1 21.1 0)
Vsrc 14 0 0 pwl(0 0 5.9 0 6 8 7.9 8 8 0 17 0 17.1 8 19 8 19.1 0)

VLatchComparator 16 0 pwl(0 0 5.2 0 5.3 3 5.5 3 5.6 0 10.4 0 10.5 3 10.7 3
10.8
+0 16.3 0 16.4 3 16.6 3 16.8 0 21.5 0 21.6 3 21.8 3 21.9 0)
Vsl1 3 0 5
Vsl3 13 0 5

**————BUFFERS (XB)

XB1 3 2 BUFFER
XB3 13 12 BUFFER

**————SELECT TRANSISTORS (MNS)

MNS1 1 2 4 4 NMS l=120e-9 w=150e-9
MNS3 14 12 10 10 NMS l=120e-9 w=150e-9

**————FLASH CELLS (XC)

XCA1 4 7 10 17 ISFC1

**————OPAMP (XOP)

XOP1 4 16 15 LATCOMP
```

```
**————SUBCIRCUITS (.SUBCKT)

.SUBCKT BUFFER 2 5
.MODEL NMS NMOS
.MODEL PMS PMOS
** 2=input 5=output
VCC 1 0 5
MP1 1 2 3 1 PMS
MN1 3 2 0 0 NMS
MP2 1 3 5 1 PMS
MN2 5 3 0 0 NMS
.ENDS

.SUBCKT LATCOMP 4 9 8
.MODEL NMS NMOS
.MODEL PMS PMOS
**LATCHED COMPARATOR CIRCUIT
** 4=input 9=latch 8=output
V1 1 0 2
V2 7 0 0.75
VB 6 0 2
MP1 1 9 2 1 PMS
MP2 2 8 3 1 PMS
MP3 2 3 8 1 PMS
MN1 3 4 5 0 NMS
MN2 8 7 5 0 NMS
MN3 5 6 0 0 NMS
.ENDS

.SUBCKT ISFC1 1 3 5 6
.INCLUDE 90nmbulk.pm
**Industry Standard Flash Cell - Writing(P=CHEI, E=FN)
** Ifn = A Epow2 exp( -B/A )
** Ichei = C D F exp( -y/D )
**1=drain 3=controlgate 5=source 6=floatinggatevoltage
Cfn 2 0 150n
Cchei 4 0 150n
Cbitline 1 0 1u
MNflash 1 6 5 0 nmos l=120e-9 w=150e-9 vth=0.9
Gcheisimplified 2 0 VALUE = ' 3e-17 * ((V(6)-V(1)) / 10e-9)
+* EXP( - 1 / ((V(6)-V(1)) / 10e-9) ) '
Gfn 4 0 VALUE = ' SGN(V(6)-V(5)) * 4.95e-25 * ( (V(6)-V(5)) / 10e-9 )
+* ( (V(6)-V(5)) / 10e-9 )* EXP(- 4.47e-7 / ( (V(6)-V(5)) / 10e-9 )) '
Esum 6 0 VALUE = ' V(3) + (4*V(2)) + (2*V(4)) '
```

```
.ENDS

.SUBCKT ISFC2 1 3 5 4
.INCLUDE 90nmbulk.pm
**Industry Standard Flash Cell - Writing(P=FN, E=FN)
** Ifn = A Epow2 exp( -B/A )
**1=drain 3=controlgate 5=source 6=floatinggatevoltage
Cfn 2 0 150n
Cbitline 1 0 1u
MNflash 1 4 5 0 nmos l=120e-9 w=150e-9 vth=0.9
Gfn 2 0 VALUE = ' SGN(V(4)-V(1)) * 4.95e-25 * ( (V(4)-V(1)) / 10e-9 )
+* ( (V(4)-V(1)) / 10e-9 ) * EXP(- 4.47e-7 / ( (V(4)-V(1)) / 10e-9 )) '
Esum 4 0 VALUE = ' V(3) + (3*V(2)) '
.ENDS

**———————————-
.op
.tran 0.01 24
.options post
.ic v(4)=0
.end
```

B Net list: 2x2 bit NOR

```
** Flash Representative (decoder, flash cell(s), sense apmlifier)
** 1 bit memory model (writing schema(s): ISFC1)

**————SOURCES (V)

**Control Gate voltage for programming and reading
Vcg1 1 0 pwl(6 0 6.1 3 8 3 8.1 0 10 0 10.1 1 12 1 12.1 0
+26 0 26.1 1 28.1 1 28.2 0)
Vcg2 2 0 pwl(14 0 14.1 1 16 1 16.1 0
+22 0 22.1 3 24.1 3.5 24.2 0 30 0 30.1 1 32 1 32.1 0)
Vcg3 3 0 pwl(6 0 6.1 3 8 3.5 8.1 0 14 0 14.1 1 16 1 16.1 0
+30 0 30.1 1 32 1 32.1 0)
Vcg4 4 0 pwl(10 0 10.1 1 12 1 12.1 0
+22 0 22.1 3 24.1 3 24.2 0 26 0 26.1 1 28 1 28.1 0)

**Bitline voltage for programming and reading
Vbtl1 10 0 pwl(6 0 6.1 3 8 3 8.1 0 10 0 10.1 3 12 3 12.1 0
+14 0 14.1 3 16 3 16.1 0 22 0 22.1 3 24 3 24.1 0 26 0 26.1 3 28
+3 28.1 0 30 0 30.1 3 32 3 32.1 0)
Vbtl2 12 0 pwl(6 0 6.1 3 8 3 8.1 0 10 0 10.1 3 12 3 12.1 0 14
+0 14.1 3 16 3 16.1 0 22 0 22.1 3 24 3 24.1 0 26 0 26.1 3 28 3 28.1
+0 30 0 30.1 3 32 3 32.1 0)

**Source voltage for erasing
Vsrc1 7 0 pwl(2 0 2.1 3 4.1 3 4.2 0
+18 0 18.1 3 20.1 3 20.2 0)
Vsrc2 9 0 pwl(2 0 2.1 3 4.1 3 4.2 0
+18 0 18.1 3 20.1 3 20.2 0)

**Latch for the comparator that senses the logic level
VLatchComp 90 0 pwl(9 0 9.1 3 9.6 3 9.7 0 12.4 0 12.5 3 13.0 3 13.1
+0 16.4 0 16.5 3 17 3 17.1 0 25 0 25.1 3 25.6 3 25.7 0 28.1 0 28.2 3 28.7 3
+28.8 0 32 0 32.1 3 32.6 3 32.7 0)

**————BUFFERS (XB, V)

Vcc1 100 0 5
XB1 7 6 100 BUFFER
```

```
XB2 9 8 100 BUFFER

Vcc2 101 0 1.5
XB3 10 5 101 BUFFER
XB4 12 11 101 BUFFER

**————FLASH CELLS (XC)

XCA1 5 1 6 1001 ISFC1
XCA2 11 4 6 1002 ISFC1
XCB1 5 2 8 1003 ISFC1
XCB2 11 3 8 1004 ISFC1

**————OPAMP (XOP)

**EVA 13 0 VALUE = ' SGN(V(5) - 0.5) + 1 '
**EVB 14 0 VALUE = ' SGN(V(11) - 0.5) + 1 '

XOP1 5 13 90 LATCOMP
XOP2 11 14 90 LATCOMP

**————SUBCIRCUITS (.SUBCKT)

.SUBCKT BUFFER 2 5 1
.MODEL NMS NMOS
.MODEL PMS PMOS
** 2=input 5=output 1=Vcc
MP1 1 2 3 1 PMS
MN1 3 2 0 0 NMS
MP2 1 3 5 1 PMS
MN2 5 3 0 0 NMS
.ENDS

.SUBCKT LATCOMP 4 8 9
.MODEL NMS NMOS
.MODEL PMS PMOS
**LATCHED COMPARATOR CIRCUIT
** 4=input 8=output 9=latch
V1 1 0 2
V2 7 0 0.5
VB 6 0 2
MP1 1 9 2 1 PMS
MP2 2 8 3 1 PMS
MP3 2 3 8 1 PMS
MN1 3 4 5 0 NMS
```

```
MN2 8 7 5 0 NMS
MN3 5 6 0 0 NMS
.ENDS

.SUBCKT ISFC1 1 3 5 6
.INCLUDE 90nmbulk.pm
**Industry Standard Flash Cell - Writing(P=CHEI, E=FN)
** Ifn = A Epow2 exp( -B/A )
** Ichei = C D F exp( -y/D )
** 1=drain 3=controlgate 5=source 6=floatinggatevoltage
Cfn 2 0 150n
Cchei 4 0 150n
Cbitline 1 0 1u
MNflash 1 6 5 0 nmos l=120e-9 w=150e-9 vth=1
Gcheisimplified 2 0 VALUE = ' 3e-17 * ((V(6)-V(1)) / 10e-9)
+* EXP( - 1 / ((V(6)-V(1)) / 10e-9) ) '
Gfn 4 0 VALUE = ' SGN(V(6)-V(5)) * 4.95e-25 * ( (V(6)-V(5)) / 10e-9 )
+* ( (V(6)-V(5)) / 10e-9 ) * EXP(- 4.47e-7 / ( (V(6)-V(5)) / 10e-9 )) '
Esum 6 0 VALUE = ' V(3) + (4*V(2)) + (2*V(4)) '
.ENDS

**-----------------------
.op
.tran 0.01 34
.options post
.ic v(5)=0
.ic v(11)=0
.end
```

C
Net list: 2x1 bit NAND

```
** Flash Representative (decoder, flash cell(s), sense apmlifier)
** 1 bit memory model (writing schema(s): ISFC2)

.MODEL NMS NMOS

**————SOURCES (V)

**Control Gate voltage for programming and reading
Vcg1 1 0 pwl(2 0 2.1 3 4 3 4.1 0 10 0 10.1 1 12 1 12.1
+0 14 0 14.1 3 16 3 16.1 0)
Vcg2 2 0 pwl(2 0 2.1 3 4 3 4.1 0 6 0 6.1 3 8 3 8.1 0 10
+0 10.1 3 12 3 12.1 0 14 0 14.1 1 16 1 16.1 0)

**Bitline voltage for programming and reading
Vbtl 6 0 pwl(6 0 6.1 5 8 5 8.1 0 10 0 10.1 2 12 2 12.1 0
+14 0 14.1 2 16 2 16.1 0)

**Latch for the comparator that senses the logic level
VLatchComp 90 0 pwl(9 0 9.1 3 9.6 3 9.7 0 12.4 0 12.5 3 13.0
+3 13.1 0 16.4 0 16.5 3 17 3 17.1 0)

Vb1 5 0 3

**————SELECT (MNS)

MNS1 6 5 4 4 NMS l=120e-9 w=150e-9

**————FLASH CELLS (XC)

XCA1 3 1 0 1001 ISFC2
XCA2 4 2 3 1002 ISFC2

**————OPAMP (XOP)

XOP1 4 7 90 LATCOMP

**————SUBCIRCUITS (.SUBCKT)
```

```
.SUBCKT BUFFER 2 5 1
.MODEL NMS NMOS
.MODEL PMS PMOS
** 2=input 5=output 1=Vcc
MP1 1 2 3 1 PMS
MN1 3 2 0 0 NMS
MP2 1 3 5 1 PMS
MN2 5 3 0 0 NMS
.ENDS

.SUBCKT LATCOMP 4 8 9
.MODEL NMS NMOS
.MODEL PMS PMOS
**LATCHED COMPARATOR CIRCUIT
** 4=input 8=output 9=latch
V1 1 0 2
V2 7 0 0.8
VB 6 0 2
MP1 1 9 2 1 PMS
MP2 2 8 3 1 PMS
MP3 2 3 8 1 PMS
MN1 3 4 5 0 NMS
MN2 8 7 5 0 NMS
MN3 5 6 0 0 NMS
.ENDS

.SUBCKT ISFC2 1 3 5 4
.INCLUDE 90nmbulk.pm
**Industry Standard Flash Cell - Writing(P:FN, E:FN)
** Ifn = A Epow2 exp( -B/A )
** 1=drain 3=controlgate 5=source 6=floatinggatevoltage
Cfn 2 0 150n
Cbitline 1 0 1u
MNflash 1 4 5 0 nmos l=120e-9 w=150e-9 vth=1
Gfn 2 0 VALUE = ' SGN(V(4)-V(1)) * 4.95e-25 * ( (V(4)-V(1)) /
+10e-9 ) * ( (V(4)-V(1)) / 10e-9 ) * EXP(- 4.47e-7 / ( (V(4)-V(1)) / 10e-9 )) '
Esum 4 0 VALUE = ' V(3) + (3*V(2)) '
.ENDS

**________________________
.op
.tran 0.01 18
.options post
.ic v(6)=0
.end
```

Bibliography

[1] Al-Ars, *Analysis of the space of functional fault models and its application to embedded drams*, Master's thesis, TU Delft, 1999.

[2] Baker, *Cmos circuit design, layout and simulation*, Wiley, 1997.

[3] Bez et al., *Introduction to flash memories*, (Proceedings of the IEEE vol. 91, no. 4, 2003).

[4] Cappelletti et al., *Flash memories*, Kluwer Academic, 1998.

[5] Dutton et al., *Electronic design automation for integrated circuits handbook, vol ii, chapter 25*, CRC Press, 2006.

[6] Hamdioui et al., *Future challanges in memory testing*, (TU Delft, 2003).

[7] Horng et al., *A realistic fault model for flash memories*, (IEEE - 1081-7735/00, 2000).

[8] Inoue et al., *Nand flash applications*, (Toshiba America Electronic Components Inc., 2004).

[9] Kang et al., *A simple flash memory cell model for transient circuit simulation*, (IEEE Electron Device Letters, vol. 26, no. 8, 2005).

[10] Laffont et al., *A new floating gate compact model applied to flash memory cell*, (Elsevier, 2003).

[11] Larcher et al., *Floating gate devices: Operation and compact modeling*, Kluwer Academic, 2003.

[12] Lee et al., *A 32-gb mlc nand flash memory with vth endurance enhancing schemes in 32 nm cmos*, (IEEE, 2011).

[13] Mauroux et al., *A two-layer spice model of the atmel tstac eflash memory technology for defect injection and faulty behavior prediction*, (IEEE - 978-1-4244-5833-2/10, 2010).

[14] Snow et al., *Fowler-nodheim tunneling into thermally grown sio2*, (Journal of Applied Physics, 1969).

[15] Ginez, *Fault modeling and testing of flash memories*, Ph.D. thesis, University of Montpellier, 2007.

[16] Goor, *Testing semiconductor memories: theory and practice*, Comtex Publishing, 1998.

[17] Hambley, *Electronics: A top-down approach to computer-aided circuit design*, Prentice Hall, 1994.

[18] Itoh, *Vlsi memory chip design*, Springer, 2001.

[19] Maloberti, *Analog design for cmos vlsi systems*, Springer-Verlag, 2001.

[20] Mohammad, *Flash memory disturb faults: Modeling simulation and test*, Ph.D. thesis, University of Wisconsin-Madison, 2002.

[21] Varhid, *Embedded systems design: A unified hardware/software approach*, Wiley, 2001.