3D FLASH MEMORY CUBE DESIGN UTILIZING COTS FOR SPACE

A Dissertation
Presented to
The Academic Faculty

By

Da Eun Shim

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2019

3D FLASH MEMORY CUBE DESIGN UTILIZING COTS FOR SPACE

Approved by:

Dr. Lim, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mukhopadhyay
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Raychowdhury
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: April 24, 2019

Only those who will risk going too far can possibly find out how far one can go.

T. S. Eliot

To my ever supportive parents and brother

ACKNOWLEDGEMENTS

I would first and foremost like to thank my current advisor Dr. Sung Kyu Lim for his great guidance and opportunities he provided me with. This thesis would not be possible without the opportunities and support from him. I feel truly lucky to have such an intelligent, supportive and enthusiastic scholar as my advisor and mentor. I look forward to continue working with him for the next couple of years. I am also thankful to Dr. Saibal Mukhopadhyay and Dr. Arijit Raychowdhury for serving as committee members for my thesis.

The project that this thesis takes part of would not have been possible without the generous funds of NASA and Irvine Sensors. I would like to thank all of my sponsors for the project including James Yamaguchi, Christian Krutzik, and Dan Nakamura.

My personal thanks goes to my colleagues of the GTCAD lab: Kyungwook Chang, Bon Woong Ku, Rakesh Perumal, Heechun Park, Jinwoo Kim, Anthony Agnesina, Sai Pentapati, Jee Hyun Lee, Yi-Chen Lu, Lingjun Zhu, Gauthaman Murali, Chengjia Shao, and Lennart Bamberg. They are truly supportive and great people to work with.

My special thanks goes to my ever supportive parents and brother. I would not have been able to come this far in my studies if it was not for my father, my forever favorite mentor, and my most supportive and loving mother. It is also very fortunate to have a brother who cares a lot about his “always” busy sister. I love each and everyone of my family and would like to dedicate this thesis to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

With the rapid growth in scope for space missions, the computing capabilities of onboard spacecraft is becoming a major limiting factor for future missions. This thesis is part of a project that designs a radiation hardened NAND Flash memory cube for use in space. Previous stacked memory cube designs focused on horizontal integration of memory dies through multiple TSVs. Using horizontal integration, however, cannot satisfy the growing desire to use commercial-off-the-shelf (COTS) dies in the space community. Thus this thesis presents a new flash memory cube design using a vertical integration method called loaf-of-bread stacking. This allows the use of COTS dies due to easy connections to the memory controller with a single TSV per die.

In addition to the stacking method, other radiation effects and the inherent NAND Flash error prone characteristics are considered in the design of the memory cube. Thus, endcaps for the Flash cube to shield radiation were designed and error mitigation methods such as error correcting codes (ECC) and scrubbing are proposed for the design. Other methods such as bad block management and wear leveling, SRIO interface are also included to make the NAND Flash cube as robust and reliable as possible.

The main focus of this thesis is on presenting the new design with all the intended features and showing some validation of the memory controller to show proof of concept for the cube design. Thus, a preliminary RTL was designed for the memory controller portion of the design and simulated results are shown in the later part of the thesis.

CHAPTER 1

INTRODUCTION

The current era shows a remarkable amount of interest in space exploration with an increase in the number of space missions. With the increase of space missions and the rapid growth in scope for each of them, the computing capabilities of onboard spacecraft and the memory capacity is becoming a major limiting factor for accomplishing future missions. Technology development to address the limit of space computing capabilities are underway with major efforts to improve central processing units (CPU) [1]. However, efforts to address the limitation of memory systems are rarely made. Recently, a 3D DRAM memory cube has been designed to improve the active memory portion of high performance space computers (HPSC) [2]. An equivalent effort should be made for the storage space for onboard systems.

1.1 Background and Motivation

The scope for space missions are growing rapidly and with the limited capacity of onboard memory systems, there are difficulties in dealing with the copious amounts of data obtained during the mission. Missions often require data-intensive operations such as terrain navigation, hazard detection and avoidance, autonomous planning and onboard science data processing. In addition to capacity improvement, the memory must be reliable and ready to operate in a radiation environment. The major goal of this thesis is to enhance the storage portion of onboard systems using Flash memory.

NOR Flash has shorter read times with the ability to randomly access cells, however, NAND Flash provides higher memory densities owing to it's smaller cells. NAND Flash is the more widely used memory in most data storage applications. Therefore, the NAND Flash is used for the design of the memory cube in this thesis.

Technology scaling has been the main method of improving NAND Flash memory for many years. However, despite these efforts by industry and academia, capacity improvement with the planar NAND Flash is slowly reaching its limits due to physical limitations. With this slowdown, industry has been looking into alternative methods to continue improving memory through 3D integration. After Samsung released the 3D V-NAND in 2012, other memory companies such as Toshiba, SK Hynix, and Micron brought their own 3D NAND products to the market one after another. These 3D NAND products stack up Flash cells vertically within the device which improve the memory capacity of NAND Flash devices without further technology scaling. Recently, Samsung has announced the mass production of their 8 tier 3D V-NAND[3]. Unfortunately, these newest products are more vulnerable to radiation effects due to the small sizes of the transistors. Smaller transistors have lower tolerances to Single Event Effects (SEE) and Single Event Functional Interrupts (SEFI) which are major considerations for any electronics meant to be used in space. Many of the electronic devices for space applications tend to be one to two generations behind the state-of-the-art technologies for this specific reason.

Despite the low radiation tolerances, there is an ongoing desire to use commercial off the shelf (COTS) dies in the space community because of it requires shorter design time and less design efforts. The use of COTS also gives the benefit of being able to use state-of-the-art devices. However, for the NAND flash memory cubes in the past, the horizontally stacked die design made it difficult to utilize COTS because it required numerous through-silicon vias (TSV). In 2014, radiation tolerant intelligent memoery stack (RTIMS) Flash was released by 3D-Plus as one of the first efforts to stack NAND Flash dies with a memory controller. This product is a package-on-package (PoP) and is a space-ready flash cube with protection against radiation. However, it can only offer up to 24Gb of memory per cube and the size of the cube could be smaller with recent technology improvements. Thus, the motivation for this thesis comes from the necessity of an NAND Flash device that can utilize COTS for the space community with improved memory capacity.

1.2 Related Work

1.2.1 Radiation Effects

As the memory cube to be designed is meant for use in space, it is important to understand the following radiation effects on electronics. The most common radiation terms used are as follows:

- Single Event Effects (SEE) : SEEs are radiation effects caused by a single particle such as protons or cosmic rays from space. SEEs can cause disrupts in the operations of electronics and thus is an important factor to consider when designing electronics for use in space. SEE rates is defined as the probability a SEE will occur within a certain given time.

- Single Event Upset (SEU) : A SEU is the case where the electronics have been affected by radiation and require either a rewrite or a reset to resume operations.

- Single Event Functional Interrupt (SEFI) : A SEFI is a special case of a SEU where the device has to go through a power reset to recover from the SEU to resume operations.

- Total Ionizing Dose (TID) : The TID value is usually the most important value that are considered for electronics meant for space. It is the cumulative degradation of a device when exposed to ionizing radiation. In the case of the design considered in this thesis, the TID is expected to be around 90krads or more [4].

With all the radiation effects that could happen in space, error correcting methods and radiation shielding would be essential for electronics meant to be used in sapce.

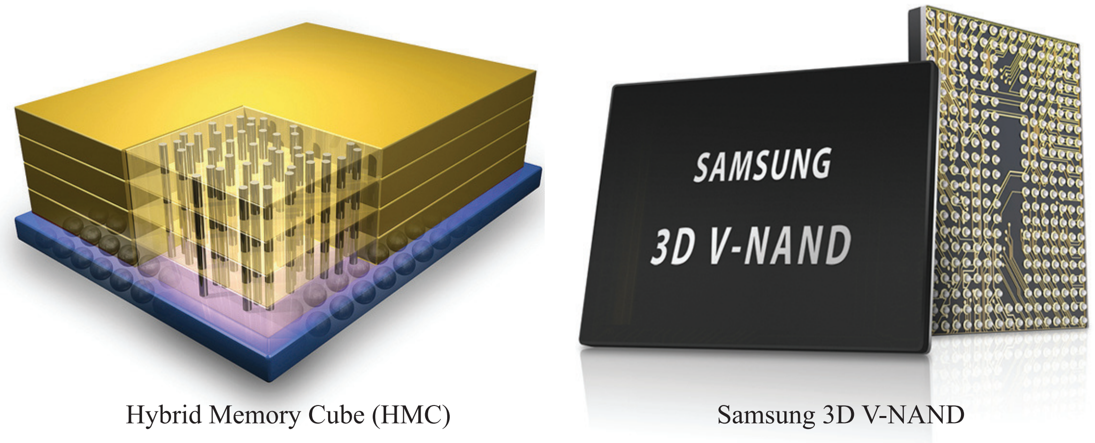
1.2.2 3D Memory cubes

Technology scaling has been slowing down in the recent years as it gradually reaches its limits with physical limitations in channel length scaling and lithographic challenges.

Global Foundries has recently abandoned the development of the 7nm technology node which is one of the indications that the end of technology scaling is not far away [5]. For this reason, industry and academia are looking into alternatives to continue improving the power consumption and performance of electronic devices. One of the most promising and major efforts is on 3D integration. Research on stacking integrated circuits with TSV connections show great potential for power and performance improvement. By stacking dies with reduced footprints instead of a single planar die, 3D ICs benefit from shorter interconnects.

Similarly, industry has been exploring the feasibility of stacking memory devices in 3D. Instead of expanding the footprint of a memory die or further decreasing the size of the transistors to increase memory capacity, companies started integrating memory in 3D. The most well known example of a 3D memory cube is the hybrid memory cube (HMC), a DRAM 3D memory cube designed by Micron. Following the advent of the HMC, the high bandwidth memory (HBM) by other competing companies have also appeared (See Figure 1.1). With the 3D integration, these memory cubes provided higher bandwidths, lower energy consumption and higher memory densities compared to the conventional 2D planar memory devices. NAND Flash is also using 3D integration to increase the memory density of a single device. As shown in Figure 1.1, Samsung has released their 3D V-NAND which announced the start of 3D NAND Flashes with many competitors following the trend. However, these 3D NAND devices are not suited for space applications due to their small transistor sizes.

In the case of memory cubes for space applications, 3D-Plus released the radiation tolerant intelligent memory stack (RTIMS) Flash in 2014 with a microprocessor based memory controller for use in space. The RTIMS is a package on package (PoP) in which two or more packages are stacked together to form a cube. Due to the nature of how PoP stacks packages instead of bare dies, PoP takes up more space and is more difficult to gain a higher memory density compared to stacking bare dies. Thus, although 3D Plus provides



Hybrid Memory Cube (HMC)

Samsung 3D V-NAND

Figure 1.1: **3D Memory Cubes. Hybrid Memory Cube [6], Samsung 3D VNAND [7]**

a space-ready flash cube with protection against radiation, the memory capacity is not that high with only 24 Gb of memory. By stacking bare dies with end cap shields and various error correction methods, it is possible to increase the memory capacity greatly while still offering similar radiation tolerances to the RTIMS.

1.2.3 Stacking Methods

The conventional way of stacking memory devices has been the “pancake” style shown on the left of Figure 1.2. Also utilized by HMC, this “pancake” method stacks dies horizontally requiring many TSVs for each die. Due to the difficulty in connecting IOs, this method limits the number of dies that can be stacked together limiting the memory density [2]. The “loaf-of-bread (LOB)” stacking method shown on the right of Figure 1.2 offers many benefits over the “pancake” method of stacking. Originally proposed by [8] [9] [10], the LOB stacking method uses vertical integration with single TSVs per die unlike the “pancake” style. The biggest advantage of the LOB configuration is that it allows the utilization of COTS dies and this is only possible because of the easy IO connections through reroute distribution layers (RDL). With the RDL, the connections come directly out of the edge through a single TSV. According to [2], LOB also has minimal electrical impedance owing to its short logic to memory interconnects, has minimal electrical impedance, and it is

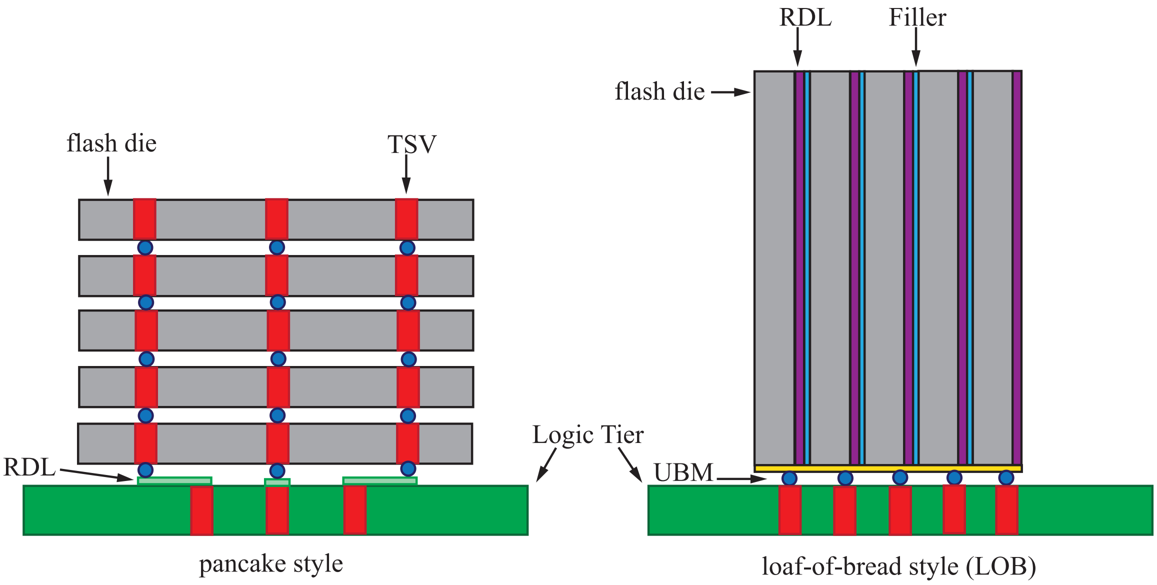


Figure 1.2: **Stacking Methods (Pancake style vs. loaf-of-bread style)**

possible to have IO connections to an individual die.

1.2.4 Re-Distribution Layer

A re-distribution layer (RDL) is an additional metal layer used to redirect IO pins or pads to the desired location. Figure 1.3 show an RDL specifically designed for a DRAM cube that was designed by Agnesina[2]. As shown in the figure, RDLs can be designed to redirect signals from one side of the metal face to another side. Having multiple layers of metals for the RDL could also improve impedance control. In the case of the memory cube design considered in this thesis, an RDL design will be especially necessary with the LOB method of stacking, as IOs need to be redirected to the bottom of each die to connect to the memory controller on the bottom.

1.3 Research Objective and Main Contribution

The research objective of the thesis is to design a reliable and robust radiation-hardened flash memory cube utilizing COTS flash dies for usage in space. This is to address the limitations of memory and computing capabilities of onboard spacecrafts resulting from

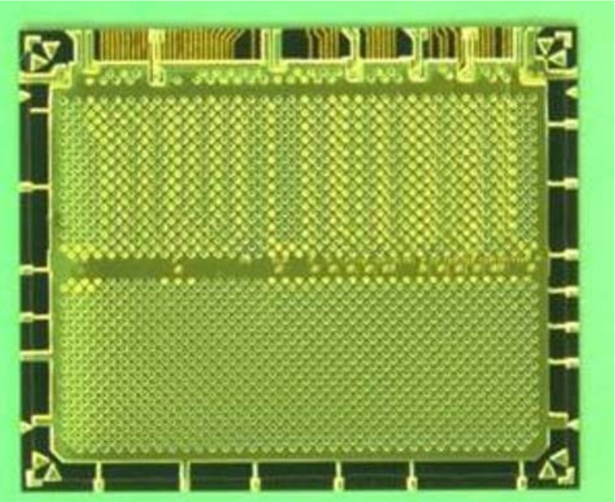


Figure 1.3: **Re-distribution Layer[2]**

the fast growth in scope for space missions. With the rapid growth, onboard computing now require memory devices with high-bandwidth, high-capacity, and high-reliability to maximize the mission data storage. Due to the radiation exposed operating environment, the memory cube resulting from this research must be robust and fault tolerant.

The main contribution of this thesis is the design of a reliable 3D NAND Flash memory cube design and the validation of the design. Thus, this thesis explores a 3D NAND flash cube design which utilizes 24 COTS dies in a LOB configuration. It will be shown that this design effectively increases data storage for onboard memory while reducing cost and design effort. In order to show proof of concept of the design, part of the memory controller is designed using RTL code. The RTL code is then simulated using the Verilog simulation model of the selected NAND Flash die.

1.4 Organization of the Thesis

The rest of this thesis is organized as follows: Chapter 2 discusses the design of the memory cube itself with discussions on the stacking methods, additional radiation shielding method, and the configuration of the cube including expected dimensions. Chapter 3 discusses the memory controller design with necessary functionalities to support the memory cube.

Chapter 4 discusses the implementation aspect of the memory controller hardware and the parts that are currently designed in register transfer level (RTL) code. Chapter 5 shows the preliminary results from simulating the memory controller RTL along with a Verilog simulation model of the selected NAND Flash die. Chapter 6 will discuss conclusions and future work.

CHAPTER 2

DESIGN OF SPACE MEMORY CUBE

2.1 Stacking Method

In the considered memory cube design, twenty four identical 32Gb NAND Flash dies are stacked together with a single memory controller connected to each die separately. In total, this offers a total memory space of 768 Gb. Each flash die has an 8-bit interface for input/output.

In order to maximize inter-connectivity and thermal performance, the loaf-of-bread (LOB) configuration first proposed by Agnesina et al [2] and which was originally used for a DRAM cube has been used for the Flash memory cube. With the die in LOB configuration, straightforward access to individual dies within the cube for power management and increased bandwidth capabilities can be achieved. It also minimizes electrical parasitics which helps reduce IO drive requirements and improves signal integrity.

NAND flash dies from multiple manufacturers have been investigated for acceptable radiation tolerance. Micron dies have been selected to be used for implementation of our cube and thus Micron die parameters will be used for comparison against the 3D Plus' RTIMS.

With the cube being in the LOB configuration, it is necessary to connect the interconnects of each die to the memory controller. In order to bring the necessary interconnections from the die I/O to the bottom edge of the die, a Reroute Distribution Layer (RDL) is applied to each die. These dies are then stacked together to form the cube structure. Figure 2.1 shows the designed RDL for this cube with the dimensions being 13.0mm by 15.5mm. With a multi-layer RDL design, both the impedance control of the design circuit and the power distribution of the die are improved. The face of the cube where the lead ex-

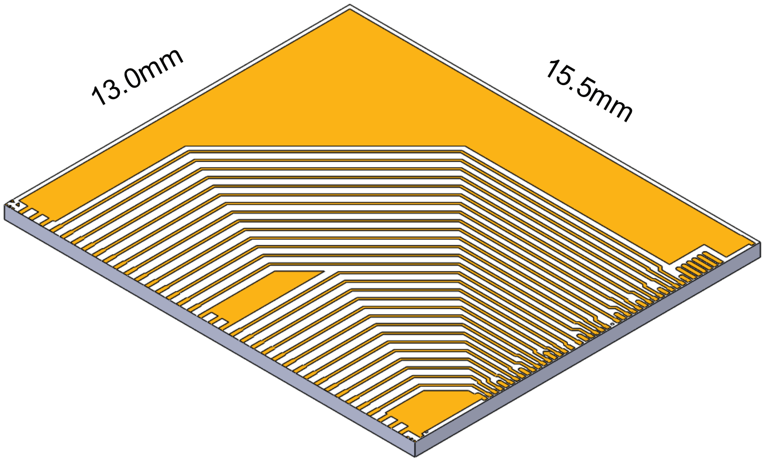


Figure 2.1: **Re-distribution Layer of the Flash cube**

tensions go to the edge will be processed with an isolation dielectric to allow for formation of interconnect pads that will subsequently be used as the I/O for the cube.

2.2 End Cap Shields

With the LOB stacking method, end caps can be added to each side of the stack for additional radiation shielding as well. The end cap adds very little additional size and weight for a stacked structure compared with a 2D layout of individual packages. Figure 2.2 shows the design of the end cap using both high-z materials to protect the cube against high energy photons such as X-rays and γ-rays and low-z materials to absorb secondary radiation from highly penetrative γ-rays. Although the end caps do not provide perfect protection, they do provide a significant reduction in the overall radiation cross-section of the device. A design tradeoff can also be made between the shielding effectiveness and the individual die thickness and number of layers.

2.3 Configuration of the Cube

With 24 NAND Flash dies vertically stacked together, the approximate dimensions of the cube are 13.8mm by 16.00mm by 15.5mm with increased memory space compared to the

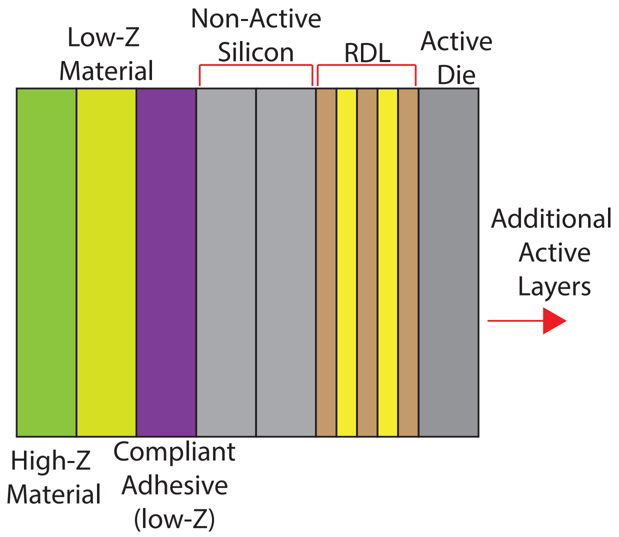


Figure 2.2: **Radiation Shield Layering for the 3D Memory Cube Ends**

RTIMS from 3D Plus. Figure 2.3 shows the Flash cube with the approximate dimensions. The configuration of the design can be seen in Figure 2.4 where 24 Flash dies are connected to the controller and each of the four dies share a capacitor layer. As mentioned earlier, each of the NAND Flash dies offer 32Gbs of data and with 24 of them, the cube will have a total of 768Gbs of memory. One page is 17,600 byes with 16,384 of data area and 1216 bytes of spare area. The ECC data and other meta data will be stored in each of the spare areas of each page. The MRAMs shown in the figure will be used for caching and Flash Translation Layer (FTL) which will be further explained in section 3.

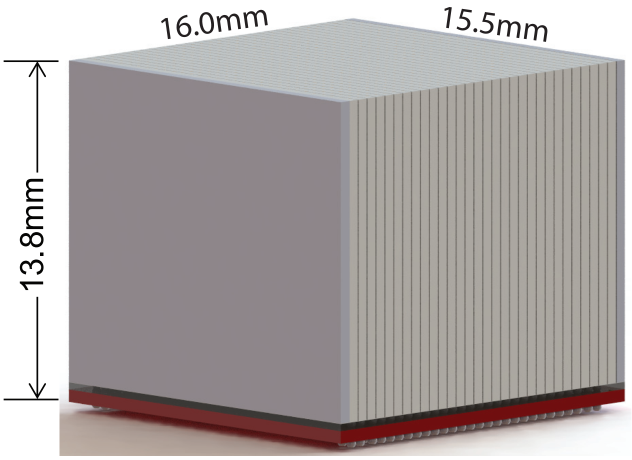


Figure 2.3: **The Flash Memory Cube**

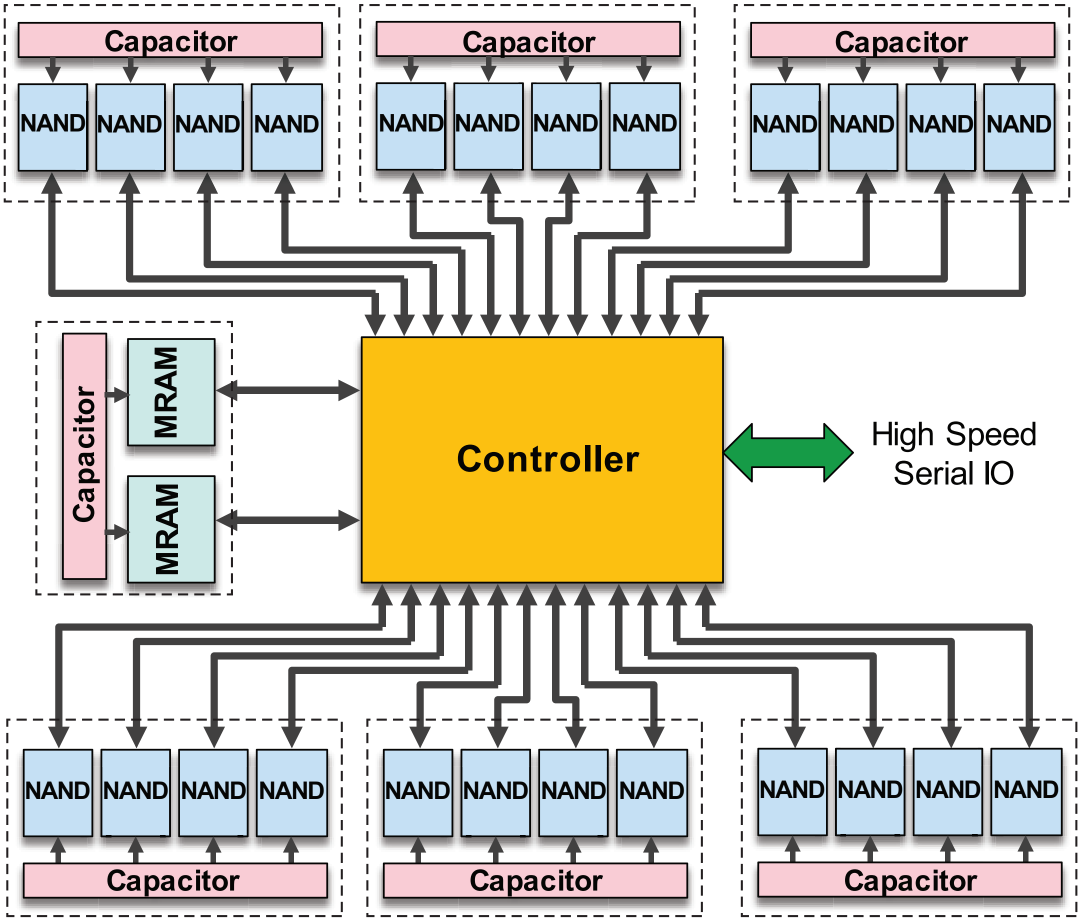


Figure 2.4: **Cube Configuration Diagram**

CHAPTER 3
DESIGN OF MEMORY CONTROLLER

In this chapter, the details for the memory controller design is demonstrated. There will be both detailed block diagrams for the design as well as figures to explain each of the features to help the understanding of the controller design.

For the memory controller of the Flash cube, a processor based implementation with firmware based control will be implemented. Figure 3.1 shows the block diagram of the whole system including both the processor subsystem and the hardware data path. The hardware data path consists of all Flash controller functions required to decouple the Flash management from the host system. The interface to the host system will consist of a high-speed serial IO in the form of daisy-chainable Serial Rapid IO (SRIO). The host will access the NAND Flash array using the SRIO interfaces as a linear-addressable memory space with no concerns of typical Flash management as the 3D memory module will handle all management functions. This will provide a simple to use interface from the host's perspective.

The NAND controller and the ECC has been implemented and is marked in Figure 3.1. The block diagram is color-coded depending on whether the module is implemented via hardware, software, or a mixture of the two.

3.1 Basic Functions

Basic read, write, erase, and reset functions has been implemented and tested. The test results of the simulated code can be found in Section 4. The interface will be to each die individually, so all the dies may be accessed at the same time, i.e. 192bit bandwidth, but not with the same set of functions. This way, each byte channel can have a different instruction. By accessing the dies at the same time, the usage of the dies can be distributed and this will

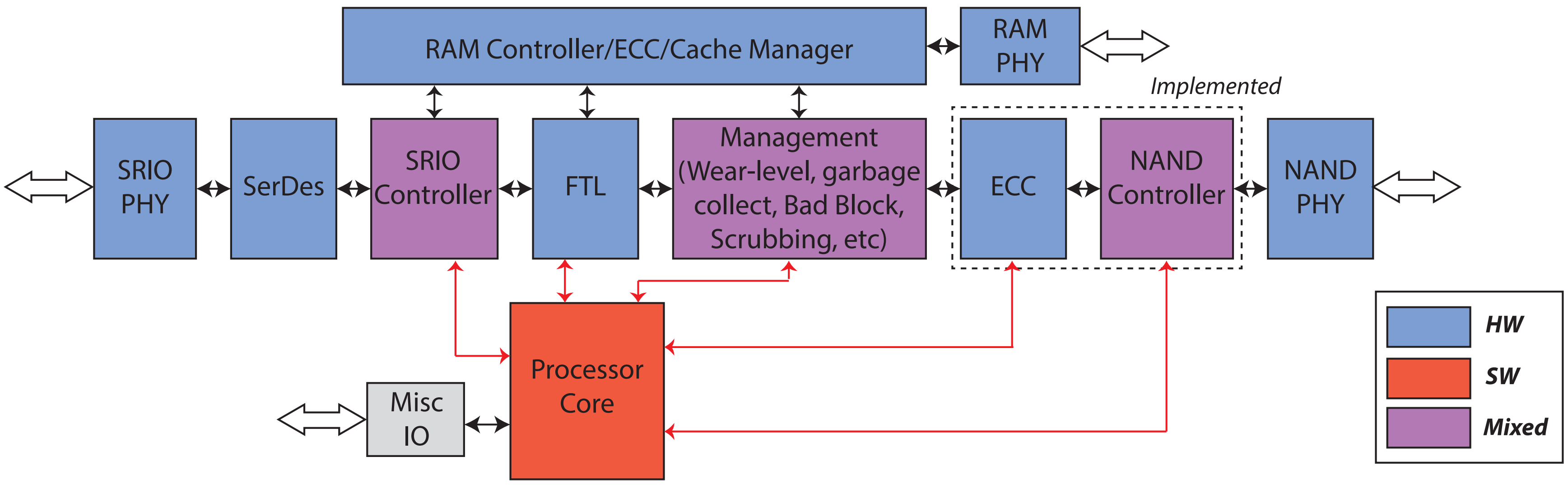


Figure 3.1: **Flash Controller Block Diagram**

also be good for wear leveling.

3.2 Error Correcting Code

In general, NAND Flash requires more control and error mitigation compared to the conventional random access memories (RAM) due to their physical properties including but not limited to bad blocks, limited program/erase cycles, having different sizes for read/write and erase. Since the memory cube is designed for use in space, it is more vital to have as much Error Correcting Code (ECC) done as possible in order to deal with the additional radiation effects on top of NAND Flash's inherent issues. For performance reasons, direct implementation of ECC in hardware is required. NAND Flash devices inherently have spare space in them which is typically used for ECC, wear leveling, and other error correcting modules. This spare space in each Flash die will mainly be used for ECC separately allowing fully parallel operation within the stack. Hamming code is not sufficient enough for a space application. Thus, in this memory cube, Bose-Chaudhuri-Hocquenghem code (BCH) is used.

3.3 Scrubbing

In addition to the basic ECC, scrubbing for the cube is used as a background task which will re-write a whole block if read errors exceed a certain threshold. The scrub rate and process will be fully programmable and the scrub algorithm will utilize the wear-leveling meta-data in order to process new address locations if block relocation is required. The baseline algorithm will be to use a combination of program/erase count and read errors to determine scrubbing action. If read errors exceed a certain threshold, the block will be re-written or relocated depending on the program/erase count. The processor will also be able to perform data movement from Flash to RAM which allows it to temporarily store the block to be relocated if required.

3.4 Bad Block Management

NAND Flash has bad blocks issues, but factory bad blocks are usually mapped out during production and do not become a big issue as they are isolated from the valid blocks. However, NAND Flash devices can also develop grown bad blocks during their lifetime that need to be mapped out of usable space when encountered. In order to resolve this issue, the controller will monitor the write status of every write as well as the maximum bit errors corrected during a read. These indicators will be used to determine if a block is ready to be retired. This function is performed on the top level directly in hardware where a faulty write will trigger a new write using the next available free block address while the failing address is stored in a small FIFO. This also requires additional logic that ties back to the write RAM buffer so that new data is not processed into the buffer until a successful write. The block address is also temporarily removed from the free list. Similarly, on each read the ECC will determine the read errors. If the errors exceed a threshold, the block address is entered into the FIFO as well with appropriate bit marker allowing various cases to be distinguished.

In addition to the hardware path, the processor core will receive an interrupt from the FIFO alerting it that there is a potential bad block. For the case of a write error, after the block has already been moved to a new location, the processor will erase the potential bad block, write a test pattern, and recheck status in the background. If the block passes, then it will be erased and returned to the free list. If it fails the second time, the block is marked as bad and added to the bad block table. Note that each block is marked bad in addition to storing a bad block table in the Flash array which provides redundancy. In the case of read error count, the processor may choose to reread the address and monitor the errors. If a potential error condition is ascertained, the processor will trigger the scrubbing module to move the block. The threshold set to the ECC module would be more conservative allowing the processor to make a final decision. The processor may then execute additional write/read of a test pattern to determine if the block should be retired. The processor code can be tuned to be more or less aggressive for bad block determination as desired.

3.5 Wear Leveling

In order to implement wear leveling, the Flash controller will be designed to store both the meta-data of total writes in the first page of each block and a SRAM based lookup table containing free blocks. The free block table will be maintained as two tables - one that holds the full free block addresses and a smaller, more optimized, partial table that hold the blocks addresses with the lowest Program/Erase count. The partial table will be maintained by the processor and will be implemented as a write-addressable FIFO in hardware in the background by scanning the Flash array for the lowest Program/Erase cycle blocks. As the processor scans the array, it will add blocks to the the free block table whenever a block has less Program/Erase cycles than the lowest Program/Erase cycle block in the table. The processor will update the table during garbage collection as well as scrubbing as needed. The partial free block table will be stored in volatile memory as it can be generated on each power-up.

In addition to the partial free block table, the full free block table is generated from the FTL (by the logical block table) as it naturally contains all available used/free blocks. However, as the FTL is addressed by logical blocks, it requires a full scan of the logical block table to generate. This will be performed at power-up by the processor. Under peak, sustained write conditions, it is possible for the partial free block table to empty in which case free blocks will be selected by the FTL in a round-robin fashion. This will incur only minimal write imbalance and can be throttled by processor if required.

3.6 SRIO Interface

Serial Rapid IO protocol will be used for the interface protocol and will be designed to support version 3.1 specifications at a minimum which added space compliance support.

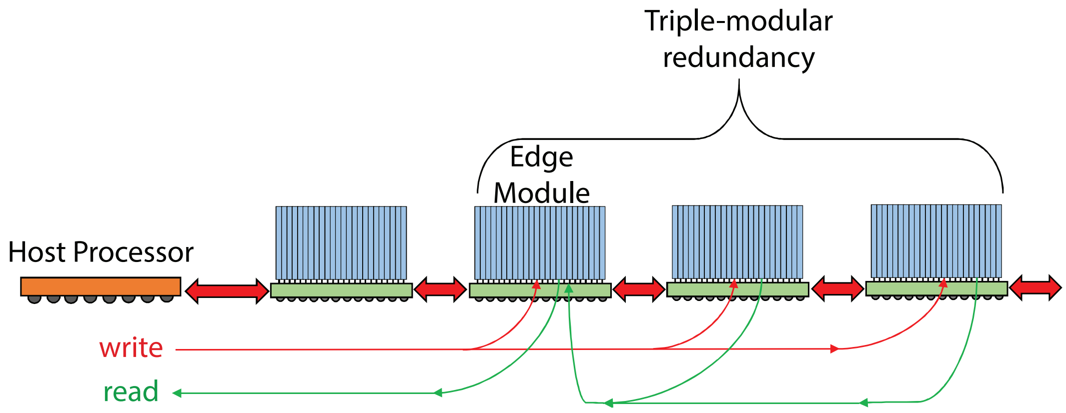


Figure 3.2: **N-Modular Redundancy Operational Diagram (TMR)**

The in/out ports will be fully buffered and the in-out path will be internally processed by the controller allowing for advanced features. The SRIO will operate at multi giga-bit speeds which will unfortunately add additional latency to each module in the daisy-chain. However, the delay can become transparent to each module in the daisy-chain with proper processing methods and since the daisy-chain works in a pipeline mode, the length of the delay should not affect the throughput. The SRIO interfaces to the wide-word interface of the 3D stack through a serializer-deserializer (SERDES) block which takes the SRIO data at Gbps rates and reduces it to a slower clock domain for processing. Since the cube includes multiple layers, the wide-word data in the lower clock domain is easily consumed by the Flash devices.

Using SRIO as the Flash IO along with daisy chain support allows the Flash controller to be designed with support for host-independent, n-modular redundancy (NMR) modes when required for increased reliability operation. Additionally, for cases where only certain data is system critical, the redundancy can be implemented within a subset of modules in a chain as shown in Figure 3.2 which shows a Triple-Modular Redundancy (TMR) technique. The redundancy can be supported in a transparent manner such that the host processor requires no additional processing or decoding. This means that a redundant configuration of Flash modules is accessed in the same manner as a single device as far as the host processor is concerned. The Flash controller would take the responsibility of passing along write operations to replicate the data among grouped modules, issuing reads to the redundant modules, and performing the voting. The device closest to the host processor in the chain is referred to as the ”edge” module as shown in Figure 3.2 and performs as the additional processor in terms of replicating and addressing additional modules, off-loading the host processor from any special duties; the ”redundant” modules are those that contain redundant data copies along with the edge module.

3.7 Flash Translation Layer

The Flash Translation Layer (FTL) will perform the logical to physical block mapping. This allows the external interface to have consistent block addressable interfaces without having to know the internal remapping operations. FTL needs fast access to the logical block table as address translation will be required for every read and write operation. A separate RAM device stores the logical block table during operation to provide fast access. The RAM device is part of the stack and will be transparent to the users.

To reduce controller complexity, the logical block table is designed as a 4-byte wide lookup table on a 16 sector (8kB) granularity. Basic operation is shown in Figure 3.3 and will be implemented in hardware. The block table stores the physical block number such that the upper bits of the sector address can be used to directly address the RAM memory to obtain the corresponding physical block. Within the physical block, the sectors will be distributed sequentially and will therefore be addressed without further translation.

The logical block table will be stored into Flash at various checkpoints during operation to reduce impact of potential unexpected power loss or other radiation effects. Redundant copies is stored across multiple dies to mitigate radiation effects and Flash block failures. For further protection, the logical block address will be stored as part of the meta-data for each page. This allows reconstruction of the block table if required. Furthermore, it provides a mechanism for the Flash controller to verify that the proper block has been addressed to reduce potential radiation induced errors on the control logic. As multiple pages are stored per block, there is an inherent level of redundancy that further protects block mapping info.

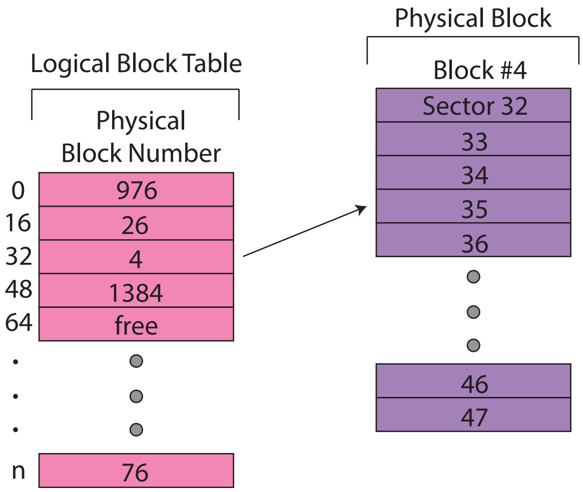


Figure 3.3: **Logical Block Table Structure**

3.8 Garbage Collection

The garbage collection will be performed as a background process. Primary garbage collection occurs via a FIFO interface from the management module and NAND controller module that keeps track when Flash blocks are relocated. The processor will interrogate the FIFO and erase those blocks in idle periods. Furthermore, the processor can scan the logical block table and Flash array to determine any erasable stale blocks that may be a result of incomplete collection due to power-down or radiation effects.

Most of this section was published in an IEEE Aerospace conference paper.

CHAPTER 4
IMPLEMENTATION

For proof of concept for the NAND Flash memory cube, the memory controller hardware has been implemented in Verilog with the simplest functionality. The Register Transfer Level (RTL) code was simulated with along with a Verilog simulation model of the Micron NAND Flash die selected. In this chapter, the implementation of the memory controller RTL code is described in detail.

4.1 Design of Hardware

The memory controller hardware design was based off of an open source NAND Flash controller developed by Lattice [11] and has been adjusted to meet the design requirements of the memory cube. The original Lattice code was written to work with a single Samsung NAND Flash die. It has been adjusted to work with twenty four of the Micron dies at the same time instead. Figures 4.1 and 4.2 are the block diagrams of the new RTL adjusted to support twenty four 32Gb Micron dies. Figure 4.1 show the overall structure with the testbench wrapper and how the Flash controller interact with the flash cube of twenty four dies. The "test case generation" block is not an actual physical block but was inserted in the figure to help understand how the testbench works. Figure 4.2 shows the modules within the flash controller module. The "logic" block within this figure is also not an actual module but represent all the necessary muxes and combinational logic circuits within the flash controller module.

The testbench was written to test the basic functionalities of the flash controller such as "reset", "program page" and "read page" for the twenty four dies at once. As shown in Figure 4.1, the flash controller and the twenty four NAND dies are simulated within the testbench wrapper. The generated test cases within the testbench simulate a NAND Flash

Table 4.1: Hardware interface signals for the flash controller

Signal	Type	Description
CLK	input	Clock
rst	input	reset
DIO	inout [191:0]	IO port to the cube (8bits each for 24 dies)
CLE	output	Command latch enable
ALE	output	Address latch enable
WE_n	output	Write enable
RE_n	output	Read enable
CE_n	output	Chip enable
Wp_n	output	Write protect
R_nB	input	Ready/Busy
cmd_code	input	command code
cmd_start	input	indicates start of a command
cmd_done	output	indicates the command is done
block_value	input	block address
page_value	input	page address
col_value	input	column address
data_to_be_written	input [191:0]	data to be written in the Flash

function issued from the CPU. Once the test case starts, the flash controller will send the control signals such as “CLE”, “RE_n” and the corresponding wires connected to the cube model will go high and low depending on the command and the cycles. The bandwidth of the chosen Micron Flash die is 8 bits and thus with twenty four of them, the IO will be 192 bits at once. Table 4.1 details the hardware interface signals for the flash controller shown in Figure 4.1.

The Flash controller module in Figure 4.2 consists of three modules: Main FSM, Timing FSM, and the module responsible for ECC. The Main FSM and the Timing FSM modules take care of the control signals and thus are shown together in the control block. When a command such as “read page” or “write page” is received from the testbench through the command_start and command_code wires, the “main FSM” will go through the states and change the “t_start” and “t_cmd” wires to the appropriate values depending on the command. The “main FSM” module takes care of each of nand flash commands while the “timing FSM” takes care of the “CLE”, “ALE”, “WE_n”, “RE_n”, “CE_n” signals for the

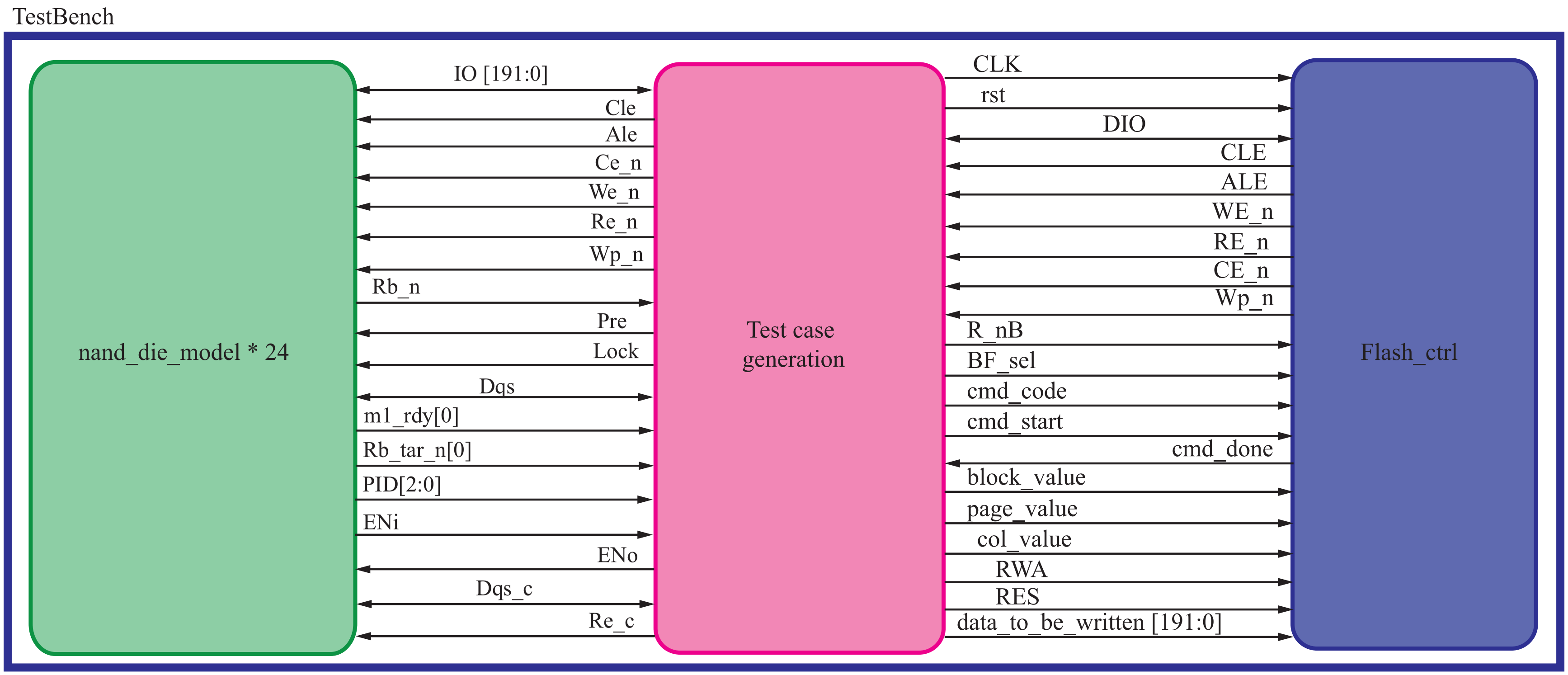


Figure 4.1: **Hardware diagram of NAND Flash controller**

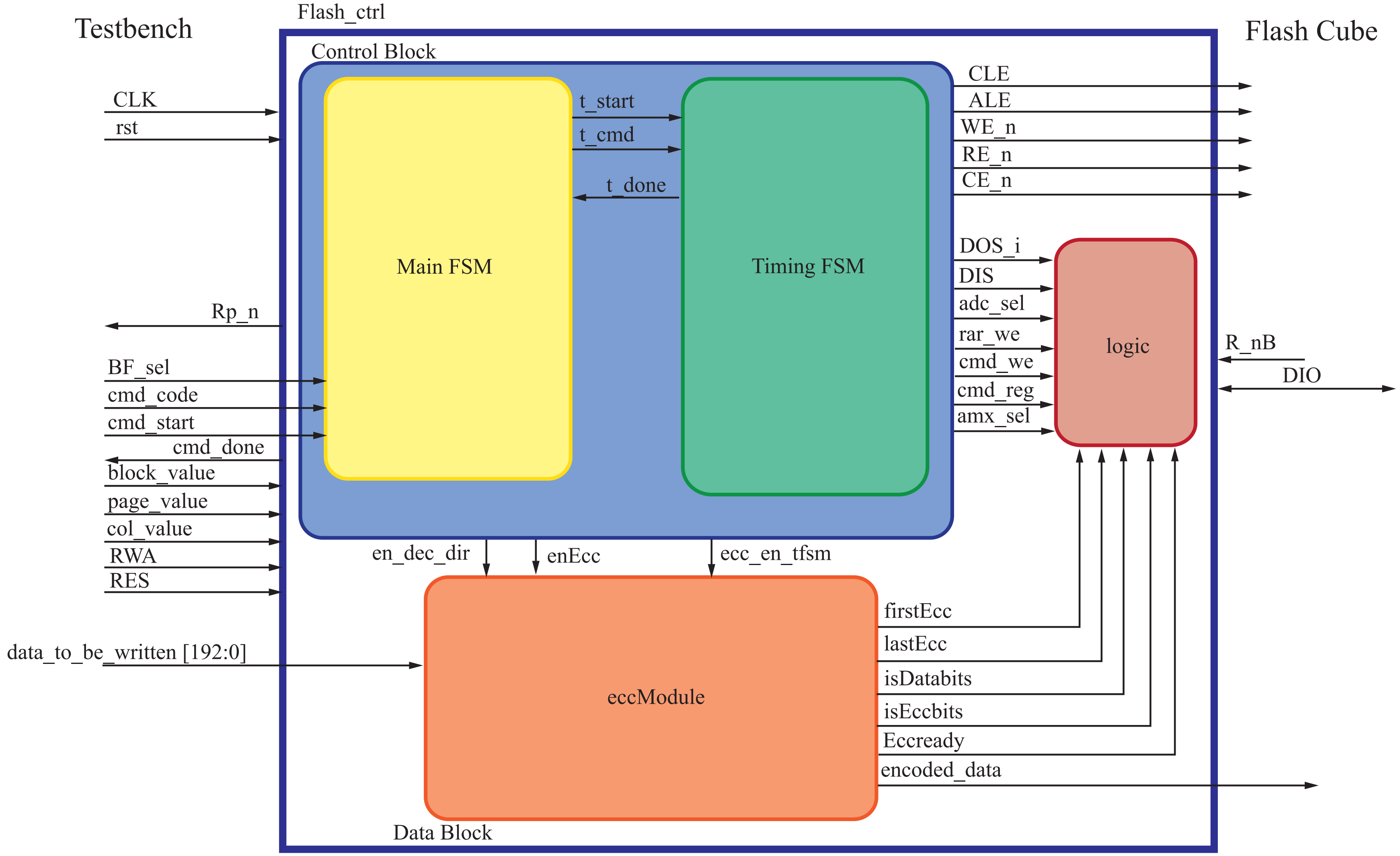


Figure 4.2: **Hardware diagram of Flash controller**

command latch, address latch and other cycles. These signals generated by the "timing FSM" module is then used by the NAND Flash simulation modules to carry out each command. The "ECC" module takes care of the BCH encoding and decoding of the data to and from the NAND Flash model. Table 4.1 details the hardware interface signals shown in Figure 4.2 excluding the signals already detailed in Table 4.1.

4.2 Implementation of Error Correction

An open source BCH code has been parameterized and used for the purpose of this implementation [12]. The original code was paramaterized to match the needs of the memory cube. In this implementation the BCH[4382, 4096] was used which corrects 22 bits per 4096 bits. With this BCH code, 286 ECC bits are created for each 4096 bits and the ECC bits would be stored in the spare area of the NAND Flash die. This would be enough to meet the requirements of the specific NAND die used for the memory cube. However, the code itself is a generic BCH and configurable if more ECC would be necessary in the future.

Figure 4.4 show the ECC block with the encoder, the decoder and the necessary parameter generator block inside. The parameter generator first takes in the "data_bits" and "target_t" signals which are 4096 and 22 respectively in the case of this design. The block calculates the necessary parameters and outputs the parameter "P". The resulting "P" looks like Figure 4.3 in this case where each 4bits have a meaning that is indicated in the Figure. Some of them are numbers inputted such as the total number of data bits to be used (4096bits in this case) and the target correction bits (up to 22bits can be corrected from this code). Others are used for calculating the syndrome and decoding. As shown in Figure 4.5, a total of 32 encoders are necessary to encode a whole page for the size of the NAND Flash die chosen.

Table 4.2: Hardware interface signals within the flash controller

Signal	Type	Description
t_start	input to Timing FSM	indicate start of timing command
t_cmd	input to Timing FSM	Timing command code
t_done	output from Timing FSM	indicate timing FSM is done with command
DOS_i	output from Timing FSM	Data out strobe
DIS	output from Timing FSM	Data in strobe
adc_sel	output from Main FSM	indicate which data should be put on to IO
rar_we	output from Main FSM	indicate whether row address should be stored
cmd_we	output from Main FSM	indicates whether command should be put to IO or not
cmd_reg	output from Main FSM	command to be put onto IO to Flash cube
amx_sel	output from Main FSM	indicate which address data should be put to IO
enEcc	input to Ecc	enable signal coming from the main FSM
ecc_en_fsm	input to Ecc	enable signal coming from the timing FSM
en_dec_dir	input to Ecc	indicate direction of Ecc (encode/decode)
firstEcc	output from Ecc	indicate that the data out is the start of encoded data
lastEcc	output from Ecc	indicate that the data out is the last of encoded data
isDatabits	output from Ecc	indicate that the data out is databits
isEccbits	output from Ecc	indicate that the data out is Ecc bits
Eccready	output from Ecc	indicate that the Ecc module is ready
encoded_data	output from Ecc	Encoded data using BCH

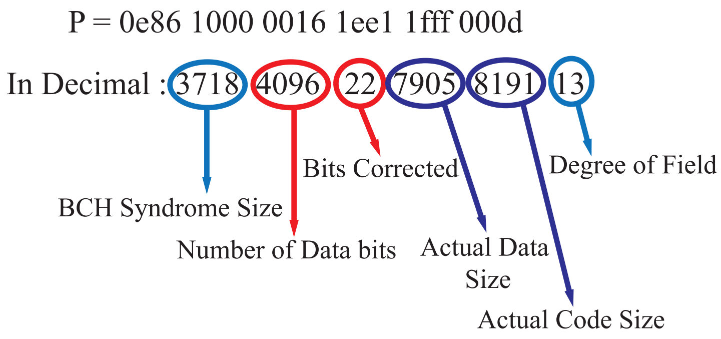


Figure 4.3: **Parameterized values for ECC module**

4.3 Functions supported

The current flash controller RTL supports the following NAND Flash commands:

- Reset : The basic reset function of NAND Flash.
- Read Page : The basic read page function of NAND Flash. The BCH decoding is embedded and for each and every page read, the data will be decoded through the controller. This is done by first encoding the data read out from the page and comparing the encoded data to the ECC data that was originally stored in the Flash for that specific data.
- Program Page : The basic program page function of NAND Flash. The BCH encoding is embedded and for each and every page is encoded when programming the data.
- Erase Block : The basic erase block function of NAND Flash. The erase block function erases a whole block of 256 pages. It is one of Flash's characteristics that it programs and read in units of pages but erases in units of blocks.
- Read ID : The basic read ID function. Each and every Flash device has an ID and this function reads the ID of the device.

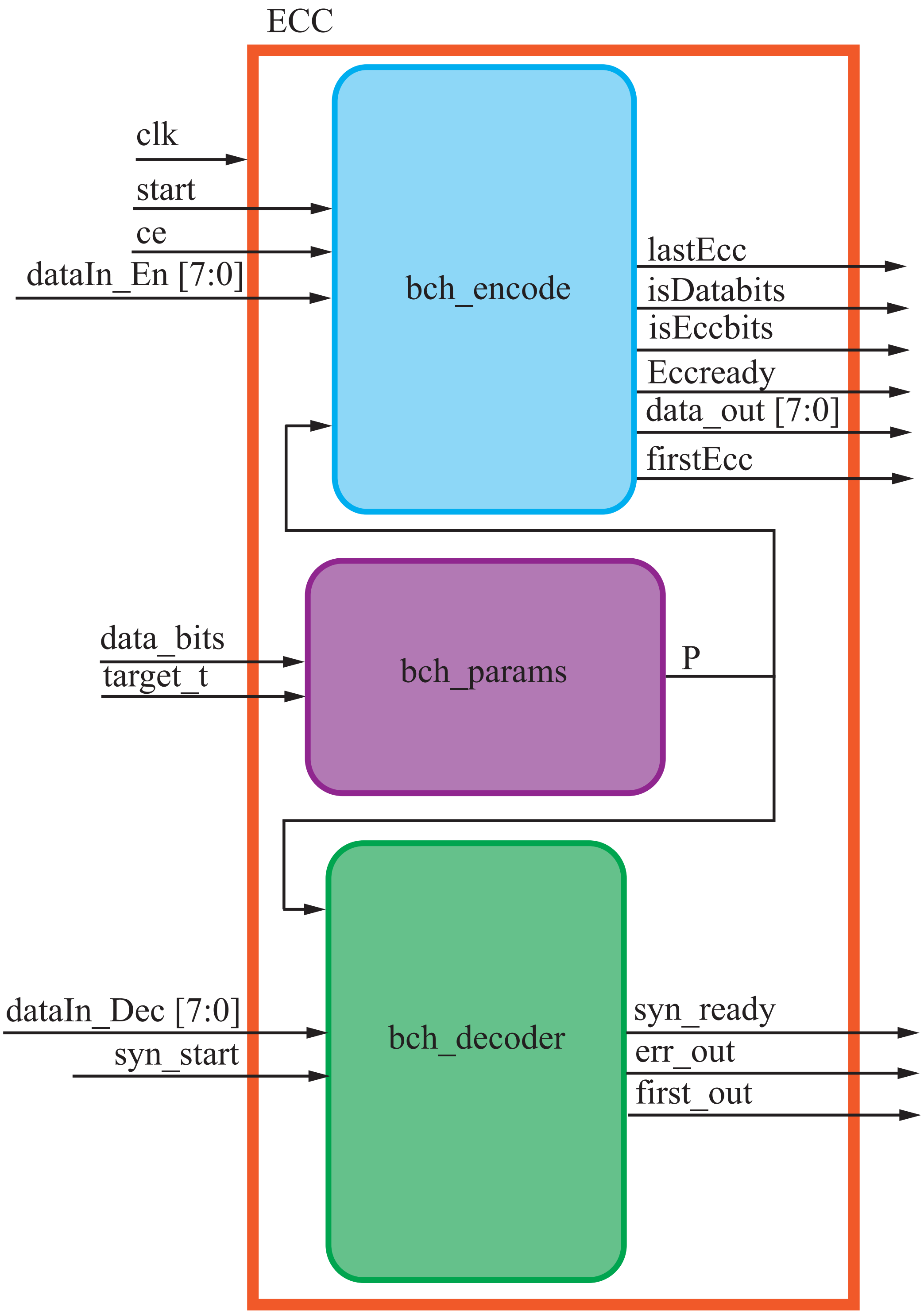


Figure 4.4: **Hardware diagram of the ECC module (BCH)**

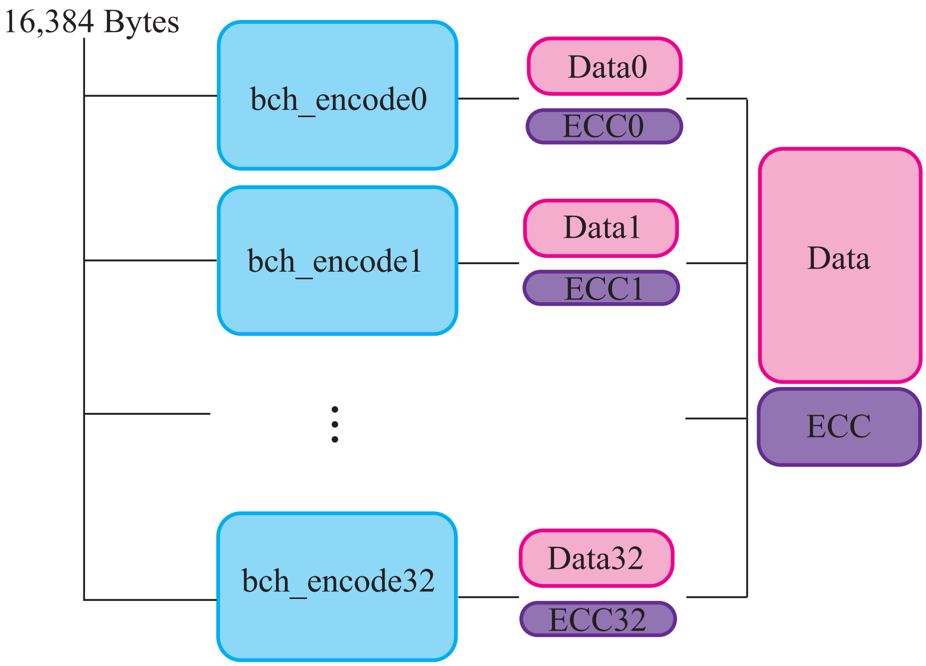


Figure 4.5: **Necessary ECCs to encode an entire page**

- Read status : The basic read status function. This gives back whether the NAND Flash device is busy or not. Due to the nature of the command, this function can be issued when the Flash is busy. This is also used as a second command in read or program functions.

CHAPTER 5
RESULTS

5.1 Verification of Code

A preliminary verification of the memory controller's RTL code has been done using a Micron Verilog model of the 32Gb NAND Flash die and the simulaiton software Vivado. Figures 5.1, 5.2, 5.3 show the simulation results of the basic functionalities of the memory controller. In Figure 5.1, the memory controller is tested for "reset", "program page", and "read page" functions. The start of each function is indicated in the waveform and it can be seen that the IO is 192bits for each cycle with 8 bit bandwidth per die and 24 NAND Flash dies. Figure 5.2 shows a more detailed waveform of a "program page" operation while Figure 5.3 shows a more detailed waveform of a "read page" operation. It is shown in the waveforms in Figures 5.2 and 5.3 that each function consists of 5 address latch cycles and the "Cle" signal goes high when starting a new function.

The waveforms shown in Figures 5.4 and 5.5 each show the encoding and decoding simulations of the BCH code. At the start of an encoding cycle, the "start" and "ce" signals go high and this is indicated by the red box in Figure 5.4. The signal boxed in yellow captures where the "first" signal goes high and this is when the first encoded data is returned through the "data_out" wires. The encoder first mirrors the input data and outputs it through the "data_out" wires. Once it is done mirroring the 4096 bits of input, it outputs 286 more bits that are the encoded ECC data. Thus, the "data_bits" and "ecc_bits" signals are necessary to indicate whether the current outputs are part of the original data bits or the encoded ECC bits. The blue box shows the cycles where the "data_bit" signal goes low and the "ecc_bits" go high indicating that the output data at those cycles are ECC data. In Figure 5.5, the decoding cycles are shown. The "errors_present" signal is low throughout the whole decoding cycle as no error was introduced to the data. The “err_count” signal is also counting “0” which is expected of the code.

5.2 FPGA Prototyping

The processor subsystem will make extensive use of the DesignStart Cortex-M0/3 processor available from ARM which provides configurable RTL for the core processor and application examples for peripheral connectivity and development. It also provides necessary tools to design and test on a simulator and then proceed with hardware prototyping using an FPGA. The first part of the RTL coding will use the MPS2+ platform as it provides ready-to-use processor evaluation designs.

After updating and testing the core configuration and peripheral connectivity on the MPS2+ platform, the design will be migrated to the selected FPGA (Xilinx FPGA) used for the 3D module Prototype system. The MPS2+ will provide a valuable reference for test and debug throughout the development. However, the MPS2+ board does not have sufficient FPGA resources to implement the necessary Flash connectivity and SRIO which is why a custom board must be developed to interface with the 3D stack.

The processor subsystem will be then further developed using a combination of the DesignStart Processor IP and custom logic circuitry. Sample projects and RTL are provided by ARM for this board and can be used to develop and debug an initial processor subsystem. Only the skeleton connectivity will be evaluated at this time. Development of the necessary peripherals including the internal bus structure and definition of the peripheral memory map will also be developed. The DesignStart includes integration tests to validate the integration in a structured way once the overall subsystem is designed.

The processor subsystem development will also include the RTL for necessary test and debug features such as trace and JTAG ports. Programming of the processor core will be performed during this task to validate end-to-end operation. This will include setup and execution of the ARM toolchain (compiler, assembler, linker, etc). The ARM Mbed

Table 5.1: **Controller Parameters at Fastest Clock Frequency**

Parameters	**Flash Controller Values**
Cell Count	6520
FFs Count	851
Wirelength	152967.6050μm
Wire Cap	8.465pF
Pin Cap	14.442pF
Switching Power	3.3031mW
Internal Power	3.59mW
Leakage Power	81.4μW
Total Power	6.706mW

cloud program may be used for initial development as it provides evaluation designs for the MPS2+. Eventually, the gcc suite of tools is preferred as it is open source and can establish a baseline development path that can be easily utilized by potential customers of the 3D module.

Similar to the Flash controller module RTL, a COTS FPGA development board, such as a Xilinx Kintex-7 KC705, will be used for hardware testing while the final FPGA based 3D module prototype board is being designed. The COTS FPGA board will be selected to match the FPGA used in the prototype board so it will be a straightforward migration.

5.2.1 Memory Controller RTL Synthesis

The RTL for the memory controller was synthesized using the NANGATE 45nm library. The fastest clock that could be achieved was 1.5ns (666.67MHz). Innovus was used to create the layout and run place and route (PnR) while Primetime was used to do static power analysis. Figure 5.6 shows the resulting layout and Table 5.1 shows the parameters for the controller when it is running on fastest clock frequency of 666.67MHz. It shows that the resulting design of the memory controller only consumes a total of $6.706mW$ of power.

5.3 Comparison to Previous products

The cube's expected radiation tolerance and memory capacity has been compared to the previous Flash cube products. The comparison of the cube's estimated radiation tolerance to the 3D plus RTIMS' radiation tolerance is shown in Table 5.2. The radiation tolerance data from [13] has been used for the purpose of the comparison due to difficulties in custom radiation testing and lack of radiation testing information from the manufacturers. It is easily seen that the memory cube design from this thesis has 16 times more memory capacity compared to the RTIMS while occupying a smaller space and providing better TID values. However, it must be noted that the radiation tolerance values used for comparison are not from direct testing of the cube and thus rigorous radiation testing must be done once the cube is manufactured.

Table 5.2: **Comparison table of FLASHRAD to RTIMS**

Parameters	3D Plus RTIMS	Our Cube : FLASHRAD
TID	>50krad (Si)	>90 krad (Si)
SEU	Immune by design	Immune by design
SEFI	Immune by design	Immune by design
Capacity	48Gb	768Gb
Dimensions	31mm*28mm*11.2mm	13.8mm*16.00mm*15.5mm

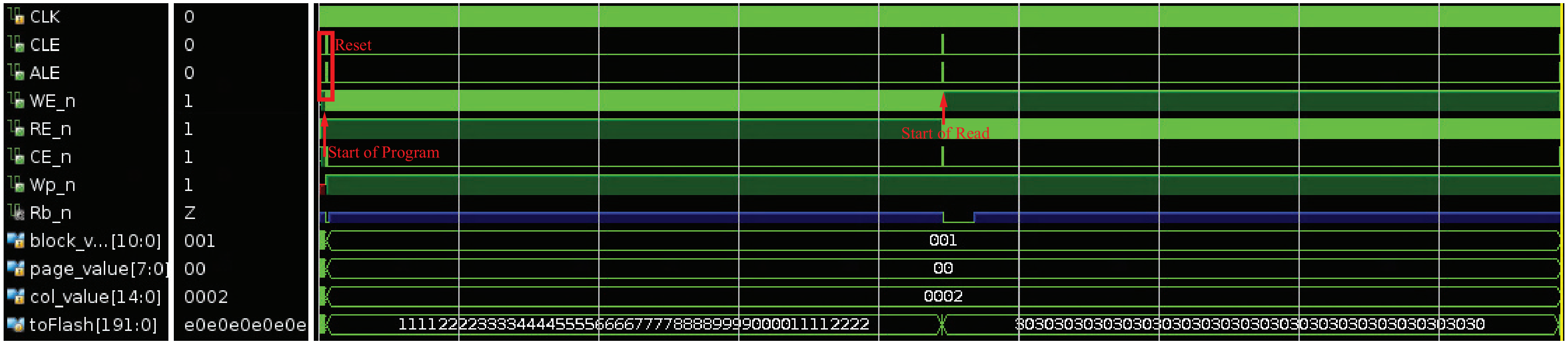


Figure 5.1: **Simulated waveform for multiple functions**

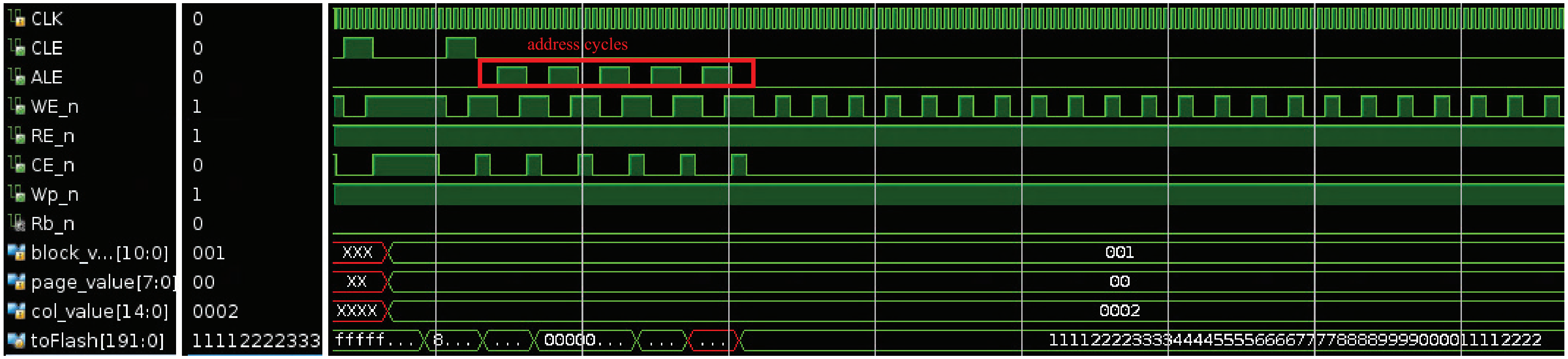


Figure 5.2: **Simulated waveform for write page operation**

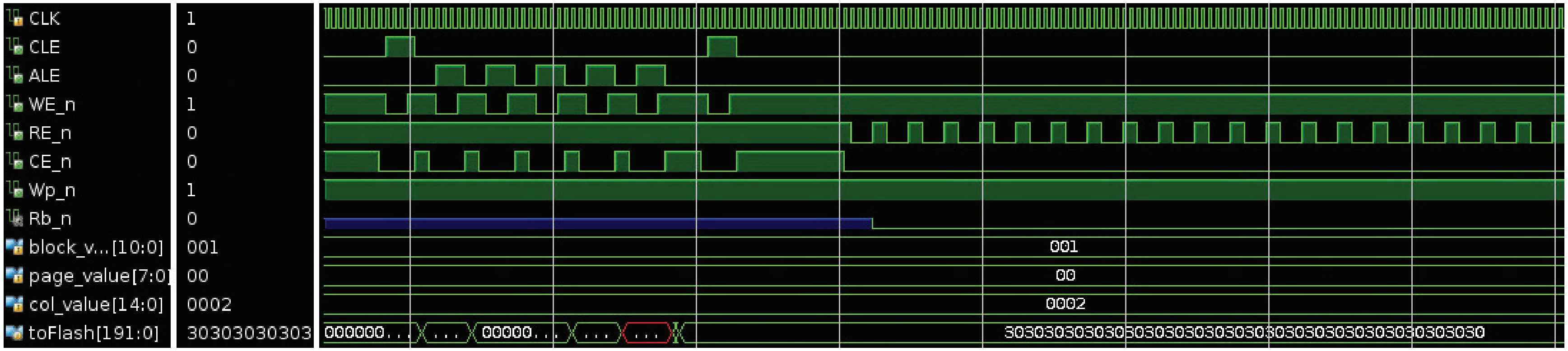


Figure 5.3: **Simulated waveform for read page operation**

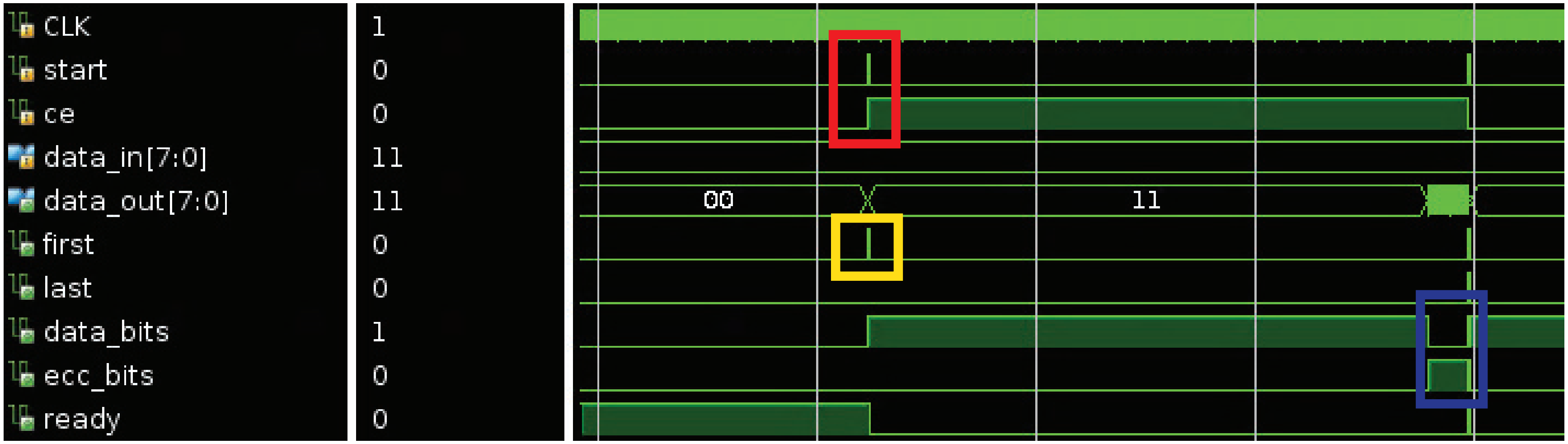


Figure 5.4: **Simulated waveform for BCH Encode**

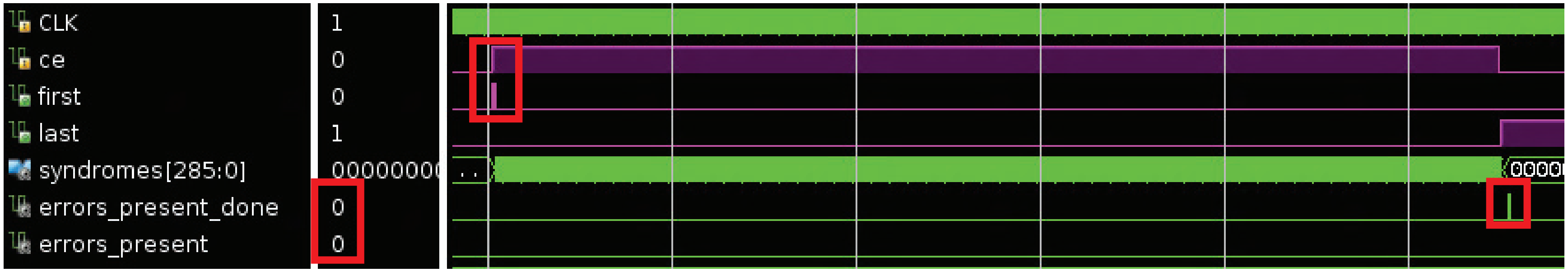


Figure 5.5: **Simulated waveform for BCH Decode**

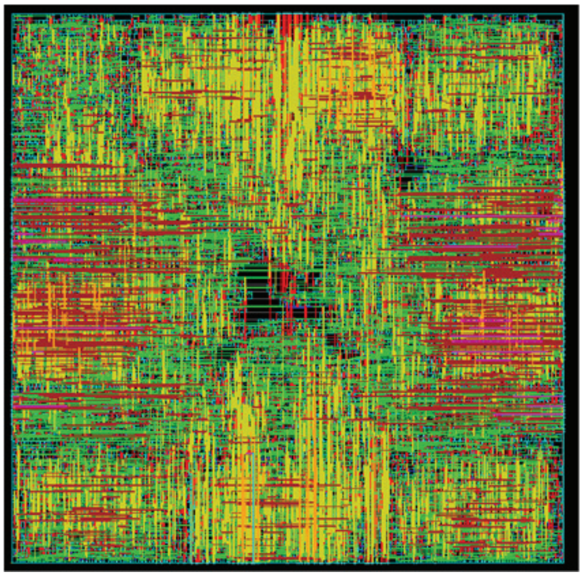


Figure 5.6: **Synthesized Memory Controller RTL**

CHAPTER 6

CONCLUSION AND FUTURE WORK

A 3D NAND Flash cube design that effectively increases memory capacity while using COTS dies to decrease cost and design effort was presented and explored in this thesis. Although further efforts will be necessary to complete and put the memory cube into use, preliminary results of the RTL simulations and ASIC design have shown promising results that indicate that the memory cube will succeed to support future space missions. Thus, the main purpose of the thesis which was to explore and validate the 3D Flash memory cube has been fulfilled.

Future works for the cube include implementation of scrubbing, bad block management, wear leveling and garbage collection as well as the flash translation layer (FTL). In addition to the additional functions for the memory controller, the fabrication of the memory cube itself is necessary. Once the cube is ready, further radiation testing would be necessary to validate the expected radiation shielding values.

REFERENCES

[1] NASA, https://gameon.nasa.gov/projects/high-performance-spaceflight-computing-hpsc/, (Accessed on 04/18/2019).

[2] A. Agnesina, A. Sidana, J. Yamaguchi, C. Krutzik, J. Carson, J. Yang-Scharlotta, and S. K. Lim, “A novel 3d dram memory cube architecture for space applications,” in *Proceedings of the 55th Annual Design Automation Conference*, ser. DAC ’18, San Francisco, California: ACM, 2018.

[3] Samsung, https://news.samsung.com/us/samsung-fifth-generation-v-nand-mass-production/, (Accessed on 04/07/2019).

[4] NASA, https://radhome.gsfc.nasa.gov/radhome/papers/seeca1.htm, (Accessed on 04/23/2019).

[5] anandtech, https://www.anandtech.com/show/13277/globalfoundries-stops-all-7nm-development, (Accessed on 04/20/2010).

[6] phys.org, https://phys.org/news/2011-12-ibm-micron-hybrid-memory-cube.html, (Accessed on 04/22/2019).

[7] EETAsia, https://www.eetasia.com/news/article/Samsung-Announces-First-512GB-Embedded-UFS?utm_source=EETI%20Article%20Alert&utm_medium=Email&utm_campaign=2019-03-01, (Accessed on 04/22/2019).

[8] C. L. Bertin, D. J. Perlman, and S. N. Shanken, “Evaluation of a 3d memory cube system,” in *Proceedings of IEEE 43rd Electronic Components and Technology Conference (ECTC)*, 1993.

[9] R. S. J.C. Carson, “Electronic module comprising a stack of ic chips each interacting with an ic chip secured to the stack,” Patent, 1994.

[10] T. Go, “Method of fabricating electronic circuitry unit containing stacked ic layers having lead rerouting,” Patent, 1991.

[11] Lattice, https://www.latticesemi.com/en/Products/DesignSoftwareAndIP/IntellectualProperty/ReferenceDesigns/ReferenceDesign04/NANDFlashController, (Accessed on 04/19/2019).

[12] Russ Dill, https://github.com/russdill/bch_verilog, (Accessed on 04/19/2019).

[13] K. Grurmann, M. Herrmann, and R. Jyvaskyla. (2014). TN-IDA-RAD-14/14 Radiation Hard Memory (RHM) Comparison between NAND-Flash and DDR3 SDRAM Test Results.