Design and Implementation of ”NAND Flash Controller”

A Project Report

submitted by

AVINASH MERUGU

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

May 2013

THESIS CERTIFICATE

This is to certify that the thesis entitled **Design and Implementation of "NAND Flash Controller"**, submitted by **Avinash Merugu**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bonafide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. V. Kamakoti
Research Guide
Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

The successful completion of this project work would not have been possible without the guidance, support and encouragement of many people. I take this opportunity to express my sincere gratitude and appreciation to them.

First and foremost, I offer my earnest gratitude to my guide, **Dr. V. Kamakoti** whose energy, knowledge and dedication has inspired me to work efficiently on the project and I thank him for motivating me, and allowing me freedom and flexibility while working on the project.

I would also like to express my sincere gratitude to Mr. G.S. Madhusudan whose persistent efforts and invaluable suggestions have been motivational and helpful. I appreciate his prolonged support and his unprecedented help throughout the project period.

My special thanks and deepest gratitude to Neel, Naveen, Suresh, Vikas and Bhaskar who have been very supportive. They have enriched the project experience with their active participation and invaluable suggestions. My utmost regards to Abhishek who has helped me design the Host interface. I appreciate his valuable inputs to the project and it has been a pleasure discussing topics and working with him.

I thank my beloved friends at IIT Madras for a great experience. I would like to thank my parents for supporting and encouraging me throughout my stay at IIT Madras and I dedicate this project to my family and friends.

ABSTRACT

KEYWORDS: ONFi, NAND Flash Controller, SSD, HDD, NVMe, PCIe

The performance gap between the processor and the disk-based storage is propelled by the physical limitations in the performance of rotating magnetic media such as Hard Disk Drives(HDDs). Storage media performance has been the bottleneck of progress for quite a time. The advent of multi-core processors has further exacerbated this performance gap. This is where Solid State Drive(SSD) comes into picture. Although SSDs are slightly costlier and provide lesser storage than HDDs, they provide a greater performance improvement as it eliminates the mechanical limitations of HDDs.

The memory being adopted in SSDs is NAND Flash Memory. Initially, certain features of NAND Flash were vendor-dependent. This made it very difficult for configuration and booting. Also, command sequences were specific to the device, making development of state machines more difficult. Standardization was clearly needed, and the first industry-accepted standard was ONFi 1.0. In order to completely utilize the performance of SSDs, a Non Volatile Memory Subsystem was designed(based on the NVM Express Specification) along with a NAND Flash Controller(based on the ONFi specification). The communication to NVMe subsystem is through PCI Express interface and the command set is based on NVMe Specification. The communication to NAND Flash Memory is through NAND Flash Controller and the command set is based on ONFi Specification. The present project deals with the design and implementation of NAND Flash Controller.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i

ABSTRACT ii

LIST OF TABLES vi

LIST OF FIGURES vii

ABBREVIATIONS viii

1 Introduction 1

1.1 Processor Architecture 1

1.1.1 Overview of Processor 2

1.1.2 My contribution 2

1.2 Introduction to NAND Flash Controller 3

1.3 Objective and Problem Statement 3

1.4 Overview of Content 4

2 Background 5

2.1 NAND Flash Attributes/Advantages 5

2.2 NAND Flash vs. NOR Flash 5

2.3 Types of NAND Flash 6

2.4 NAND Flash Applications 6

2.5 Bluespec SystemVerilog Background 7

2.5.1 Advantages of using Bluespec SystemVerilog 7

2.5.2 Overview of the BSV build process 8

2.5.3 Bluespec SystemVerilog Constructs 8

2.5.4 Application Areas of Bluespec System Verilog 9

3 NAND Flash Description **10**

3.1 NAND Flash Architecture 10

3.1.1 2 Gb NAND Flash Device Organization 10

3.1.2 16 Gb NAND Flash Array Organization per LUN 11

3.2 Storage Methods 12

3.3 LUN Functional Description 13

3.4 NAND Flash Commands 14

3.4.1 NAND Flash Caching Mode Commands 14

4 NAND Flash Controller **17**

4.1 Organization of NAND Flash Controller 17

4.1.1 Registers 17

4.1.2 Control Logic 17

4.1.3 Buffers 17

4.2 NFC Functional Description 18

4.2.1 NAND Flash Controller Block Diagram 19

4.3 Interface Signal Descriptions 20

4.3.1 Host Interface 20

4.3.2 Target(ONFi) Interface 21

4.4 NAND Flash Command Sequences 22

4.4.1 READ PAGE 22

4.4.2 PROGRAM PAGE 24

4.4.3 BLOCK ERASE 27

4.4.4 READ ID 29

4.4.5 READ STATUS 32

4.4.6 RESET 34

4.5 Verification 35

4.5.1 NAND Flash Controller Verification Setup 35

4.5.2 NAND Flash Controller Verification Test Cases 36
4.5.3 NVM Express-driven NFC Verification Setup 38
4.5.4 NVM Express-driven NFC Verification Test Cases 39

5 Conclusion and Future Work **40**

A Additional Commands **41**

B NAND Flash Controller with AXI interface **42**

LIST OF TABLES

2.1 Comparison among NAND flash devices 6

4.1 Host Interface Signals 20

4.2 Target(ONFi) Interface Signals for NAND Flash Controller 21

4.3 Target(ONFi) Interface Signals for NAND Flash Target 22

LIST OF FIGURES

1.1 Processor Architecture 1

2.1 Major Markets driving NAND Flash 7

3.1 2Gb NAND Flash Device Organization 11
3.2 16 Gb NAND Flash Array Organization per LUN 12
3.3 Typical Storage Methods 13
3.4 LUN Functional Block Diagram 14
3.5 Read Page Cache Sequential Architecture 15
3.6 Program Page Cache Architecture 16

4.1 Register Description 18
4.2 NAND Flash Controller Block Diagram 19
4.3 Read Page State Machine 23
4.4 Program Page State Machine 25
4.5 Block Erase State Machine 27
4.6 Read ID State Machine 29
4.7 Read ID Response of a Micron NAND Flash Device 31
4.8 Read Status State Machine 32
4.9 Reset State Machine 34
4.10 Verification Setup for verifying NAND Flash Controller 36
4.11 NAND Flash Controller Test Cases 37
4.12 Verification Setup for verifying NVMe-driven NAND Flash Controller . 38

B.1 NAND Flash Controller with AXI interface 42

ABBREVIATIONS

CPU	Central Processing Unit
BSV	Bluespec SystemVerilog
NFC	NAND Flash Controller
ONFi	Open NAND Flash Interface
NVMe	Non-Volatile Memory Express
PCIe	Peripheral Component Interconnect Express
LUN	Logical Unit / Logical Unit Number
ECC	Error Correction Code
SSD	Solid-State Drive
HDD	Hard Disk Drive
DRAM	Dynamic Random Access Memory
CLE	Command Latch Enable
ALE	Address Latch Enable
SLC	Single-Level Cell
MLC	Multi-Level Cell
TLC	Triple-Level Cell
DUV	Design Under Verification

CHAPTER 1

Introduction

1.1 Processor Architecture

The illustration of processor architecture is shown in figure 1.1.

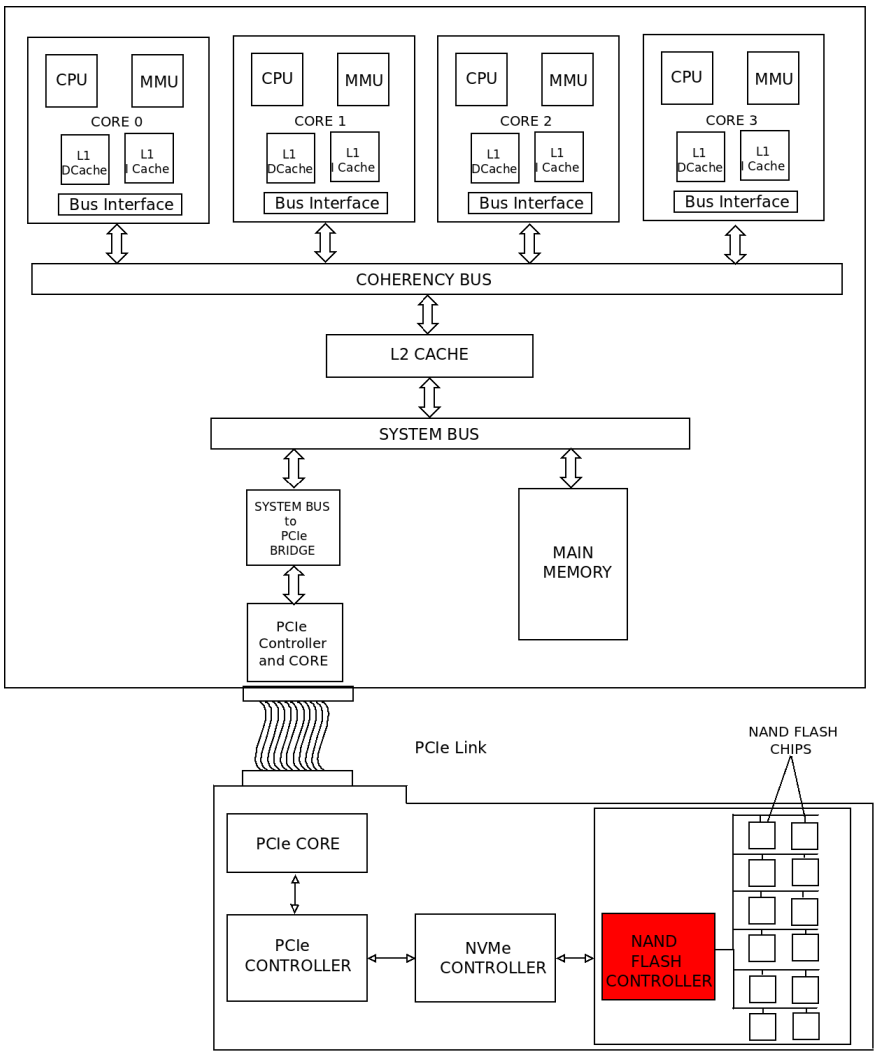

Figure 1.1: Processor Architecture

1.1.1 Overview of Processor

The processor design team in the Reconfigurable and Intelligent Systems Engineering (RISE) lab has been active in the design of a high-end in-house processor for defence and security related applications. The figure 1.1 shows an overall architecture of the expected processor. The processor system has a quad-core architecture, with support for two levels of cache hierarchy and a single on-chip DRAM. It implements cache coherency at L1 cache level, with coherency bus. The CPU core is based on 64 bit PowerPC Instruction Set Architecture. It supports dual-issue and out of order execution. A Memory Management Unit was designed for an efficient data transfer between the Processor Core and the Main Memory (DRAM). It supports NVM Express based I/O Subsytem with a PCI Express interface. This I/O subsystem typically implements the File System for the processor. NAND Flash Controller is included in the design for non-volatile memory management. It acts as a controller interface between NVM Express and NAND Flash memory. Also, it can accept commands directly from the Host processor and perform NAND Flash operations.

1.1.2 My contribution

The highlighted portion in the figure 1.1 shows my contribution in this processor design. NVM express based I/O subsystem is designed to meet the requirements of storing large amounts of data and to access Flash memory. SSDs are slowly overtaking the HDDs in the storage market and NAND Flash is being rapidly adopted for SSDs. NAND Flash Controller is designed to provide an interface between NVM Express and NAND Flash memory. It can as well provide an interface between host processor and NAND Flash memory which can be utilized to boot the system using NAND Flash.

1.2 Introduction to NAND Flash Controller

NAND Flash Controller (NFC) provides an interface for user to communicate with NAND Flash devices. The controller accepts NAND Flash Memory commands from the user interface and generates different cycles on memory interface according to the NAND Flash Memory protocol.

The beginning of NAND Flash was very trying for manufacturers of controllers and host device solutions. Interface timing, page ID location, and format typically were vendor-dependent. This made it very difficult for configuration and booting. Also, command sequences were specific to the device, making development of state machines more difficult. Many solutions at the time were software-based due to the changes among vendors. Standardization was clearly needed, and the first industry-accepted standard was ONFi 1.0.

The Open NAND Flash Interface (ONFi) is an industry Workgroup made up of more than 100 companies that build, design-in, or enable NAND Flash memory. They are dedicated to simplifying NAND Flash integration into consumer electronic products, computing platforms, and any other application that requires solid state mass storage. ONFI 2.2, the most widely used specification today, was ratified in October of 2009. The newest Open NAND Flash Interface (ONFI) 3 standard offers many benefits for high-performance NAND Flash applications and makes it easier for the designer to achieve a successful implementation. The specification was released in March 2011.

1.3 Objective and Problem Statement

- Design of ONFi-compliant NAND Flash Controller.
- Design of Host interface to be compliant with NVM express.

1.4 Overview of Content

Chapter-2 includes the background of NAND flash.

Chapter-3 includes the description of NAND flash architecture and design.

Chapter-4 includes the design and implementation of NAND flash controller.

Chapter-5 includes the conclusion and the description of future work.

CHAPTER 2

Background

2.1 NAND Flash Attributes/Advantages

- Non-volatile memory.
- High bandwidth and small access latency.
- Low power consumption.
- High chip density.

2.2 NAND Flash vs. NOR Flash

The real benefits of NAND Flash are faster PROGRAM and ERASE times, as NAND Flash delivers sustained WRITE performance exceeding 7 MB/s. Block erase times are an impressive 500s for NAND Flash compared with 1 second for NOR Flash.

The hardware pin requirements for NAND Flash and NOR Flash interfaces differ markedly. NOR Flash requires approximately 44 I/O pins for a 16-bit device, while NAND Flash requires only 24 pins for a comparable interface[1].

Clearly, NAND Flash offers several compelling advantages. The one challenge is that it is not well-suited for direct random access. This can be handled with code shadowing.

2.3 Types of NAND Flash

1. SINGLE-LEVEL CELL(SLC) NAND FLASH

   Stores 2 states per memory cell and allows 1 bit programmed/read per memory cell.

2. MULTI-LEVEL CELL(MLC) NAND FLASH

   Stores 4 states per memory cell and allows 2 bit programmed/read per memory cell.

3. TRIPLE-LEVEL CELL(TLC) NAND FLASH

   Stores 8 states per memory cell and allows 3 bit programmed/read per memory cell.

Table 2.1: Comparison among NAND flash devices

$Attribute$	SLC $(< 1Gb)$	SLC $(> 1Gb)$	MLC
Page size(Bytes)	512+16	2,048+64	4,096+128
Pages per block	32	64	128
t_R(read)	15μs(max)	20μs(max)	50μs(max)
t_{PROG}(program)	200μs(typ) 500μs(max)	200μs(typ) 700μs(max)	600μs(typ) 1,200μs(max)
t_{BERS}(erase)	2ms(typ) 3ms(max)	1.5ms(typ) 2ms(max)	3ms(typ)
Endurance cycles	100K	100K	10K

2.4 NAND Flash Applications

- USB flash drives
- Solid-state drives
- Memory cards
- Smart phones
- Ultrabooks
- Handheld audio and video players
- Digital cameras

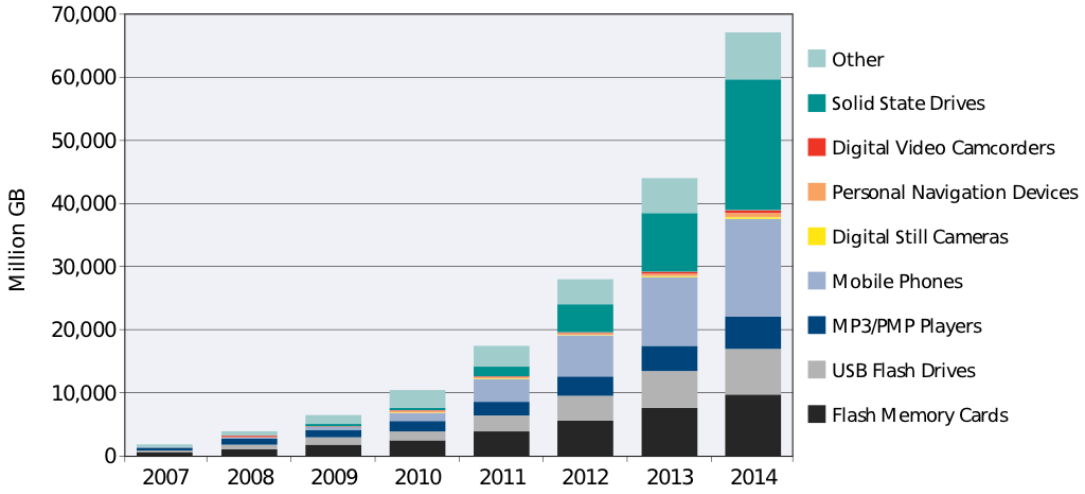

Figure 2.1: Major Markets driving NAND Flash

2.5 Bluespec SystemVerilog Background

Bluespec SystemVerilog(BSV) is a a high-level functional hardware description programming language for chip design and electronic design automation. The justification behind writing chip designs in Bluespec is that it leads to shorter, more abstract, and verifiable source code, as well as type-checked numeric code. Because of its expressive power(comparable to most advanced programming languages), full synthesizability, and quality of synthesis, Bluespec is fundamentally changing long-held assumptions about design flow.

2.5.1 Advantages of using Bluespec SystemVerilog

- Powerful parameterization and 'generate'.
- High-level of abstraction.
- Fully synthesizable at all levels of abstraction.
- Advanced clock management.
- Powerful static checking.

2.5.2 Overview of the BSV build process

The following are the steps involved in building a BSV design[2]:

1. A designer writes a BSV program. It may optionally include Verilog, SystemVerilog, VHDL, and C components.
2. The BSV program is compiled into a Verilog or Bluesim specification. This step has two distinct stages:
   - pre-elaboration - parsing and type checking
   - post-elaboration - code generation
3. The compilation output is either linked into a simulation environment or processed by a synthesis tool.

Once the Verilog or Bluesim implementation is generated, the workstation provides the following tools to help analyze your design:

- Interface with an external waveform viewer with additional Bluespec-provided annotations, including structure and type definitions.
- Schedule Analysis viewer providing multiple perspectives of a modules schedule.
- Scheduling graphs displaying schedules, conflicts, and dependencies among rules and methods.

2.5.3 Bluespec SystemVerilog Constructs

Rules: Rules are used to describe how data is moved from one state to another, instead of the Verilog method of using always blocks[3]. Rules have two components:

- Rule conditions: Boolean expressions which determine when the rule is enabled.
- Rule body: a set of actions which describe state transitions

Modules: A module consists of three things: state, rules that operate on that state, and an interface to the outside world (surrounding hierarchy). A module definition specifies a scheme that can be instantiated multiple times.

Interfaces: Interfaces provide a means to group wires into bundles with specified uses, described by methods. An interface is a reminiscent of a *struct*, where each member is a method. Interfaces can also contain other interfaces.

Methods: Signals and buses are driven in and out of modules with methods. These methods are grouped together into interfaces. There are three kinds of methods:

- Value Methods: Take 0 or more arguments and return a value.
- Action Methods: Take 0 or more arguments and perform an action(side-effect) inside the module.
- ActionValue Methods: Take 0 or more arguments, perform an action, and return a result.

Functions : Functions are simply parameterized combinational circuits. Function application simply connects a parameterized combinational circuit to actual inputs.

2.5.4 Application Areas of Bluespec System Verilog

- Modeling for Software development
- Modeling for Architecture Exploration
- Verification
- IP creation

CHAPTER 3

NAND Flash Description

3.1 NAND Flash Architecture

NAND Flash device is comprised of a number of LUNs. LUN or Die is the minimum unit that can independently execute commands and report status. A set of LUNs that share one CE(Chip Enable) signal within one NAND Flash device form a *NAND Target*[4].

NAND Flash Device is organized as *blocks*. Block consists of multiple *pages* and is the smallest addressable unit for erase operations. Page is the smallest addressable unit for read and program operations.

3.1.1 2 Gb NAND Flash Device Organization

The 2Gb NAND Flash device is organized as 2048 blocks, with 64 pages per block (see Figure 3.1). Each page is 2112 bytes, consisting of a 2048-byte data area and a 64-byte spare area. The spare area is typically used for ECC, wear-leveling, and other software overhead functions, although it is physically the same as the rest of the page.

Many NAND Flash devices are offered with either an 8- or a 16-bit interface. Host data is connected to the NAND Flash memory via an 8-bit- or 16-bit-wide bidirectional data bus. For 16-bit devices, commands and addresses use the lower 8 bits (7:0). The upper 8 bits of the 16-bit data bus are used only during data-transfer cycles.

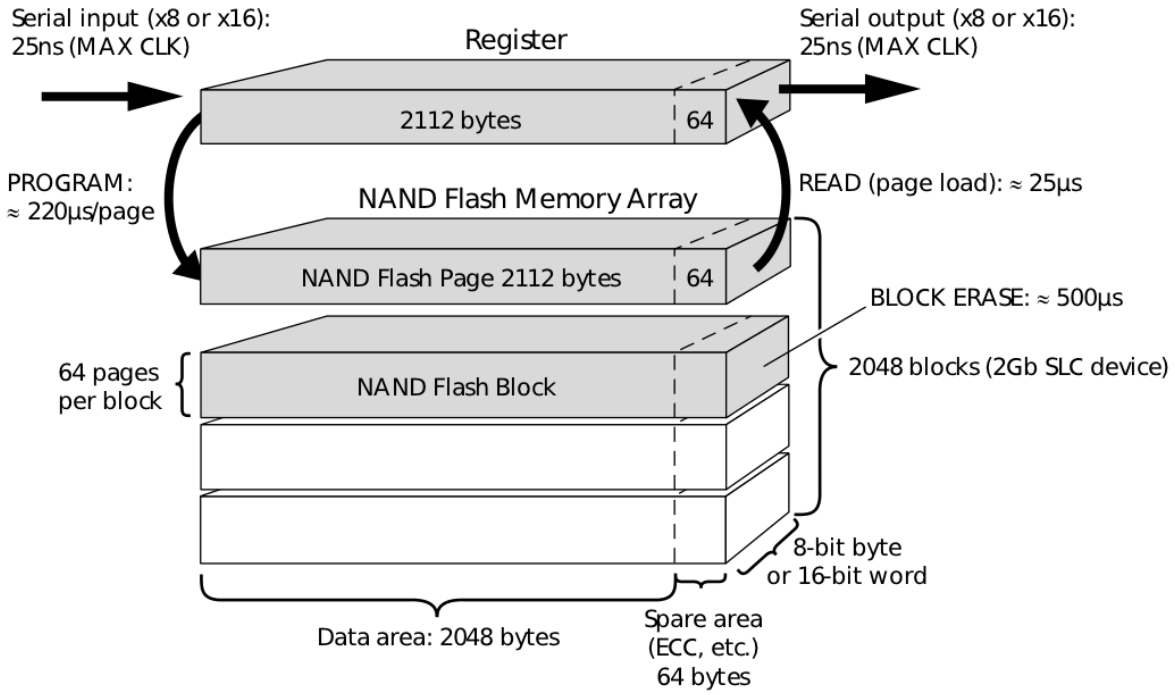

Figure 3.1: 2Gb NAND Flash Device Organization

3.1.2 16 Gb NAND Flash Array Organization per LUN

The large-block NAND flash device organization is the same as that of a small-block(2 Gb) device except for the size of each block and size of each page. Also, there can be different *planes* in a LUN. NAND Flash supports multi-plane operations which can be used to improve the performance of the device. A NAND flash device is comprised of many LUNs and so, the array organization per LUN is shown in figure 3.2[5].

There are 2 planes per LUN and each plane has 2048 blocks. So, there are a total of 4096 blocks per LUN. Each block has 128 pages with each page of size 4,320 bytes. In that total of 4,320 bytes, 4 KB is used for storage of data, while the remaining 224 bytes are used for ECC and other software functions.

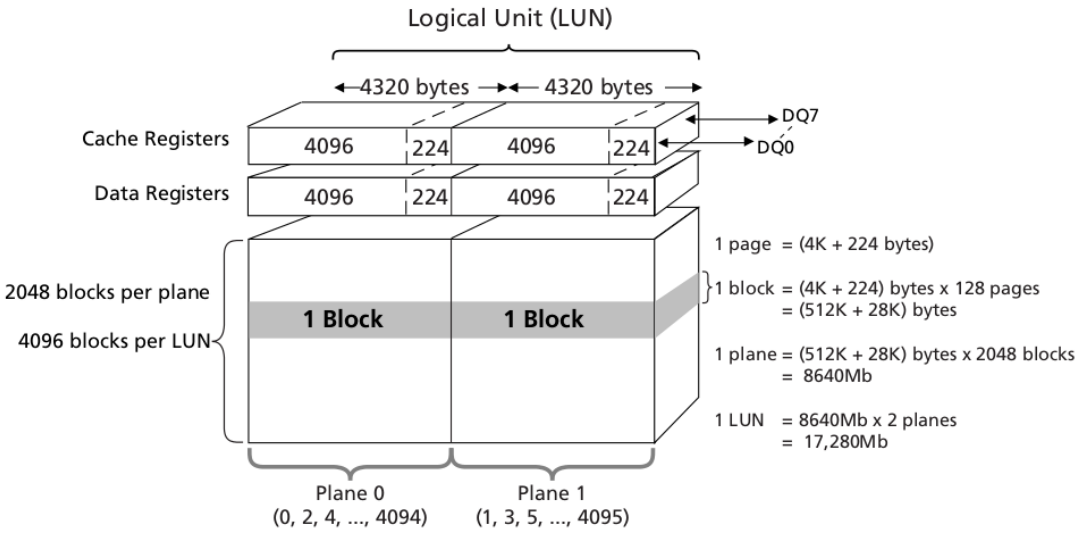

Figure 3.2: 16 Gb NAND Flash Array Organization per LUN

In case of ECC, every time a *page read* operation is called, the page data is compared with the ECC data in the spare area. If there is an error in the stored data, the NFC receives an error-interrupt. After the data is corrected, the page read process resumes.

3.2 Storage Methods

The two common methods for storing data and spare information in the same page are shown in Figure 3.3. The first method shows a data area of 512 bytes plus the 16-byte spare area directly adjacent to it; 528 bytes for the combined areas. A 2112-byte page can contain four of these 528-byte elements[1].

The second implementation involves storing the data and spare information separately. The four 512-byte data areas are stored first, and their corresponding 16-byte spare areas follow, in order, at the end of the page.

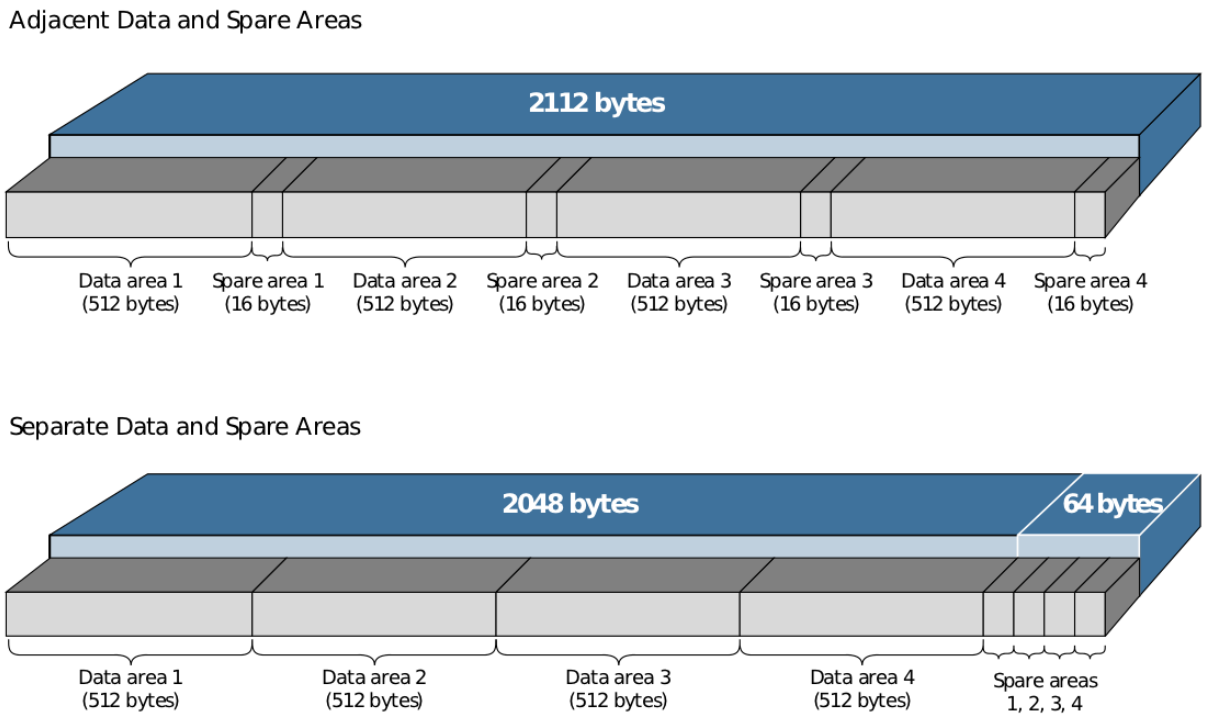

Figure 3.3: Typical Storage Methods

3.3 LUN Functional Description

NAND Flash Devices use NAND Flash electrical and command interfaces. Data, commands, and addresses are multiplexed onto the same pins and received by I/O control circuits. The commands received at the I/O control circuits are latched by a command register and are transferred to control logic circuits for generating internal signals to control device operations. The addresses are latched by an address register and sent to a row decoder to select a row address, or to a column decoder to select a column address.

Data is transferred to or from the NAND Flash memory array, through a data register and a cache register. The NAND Flash memory array is programmed and read using page-based operations and is erased using block-based operations.

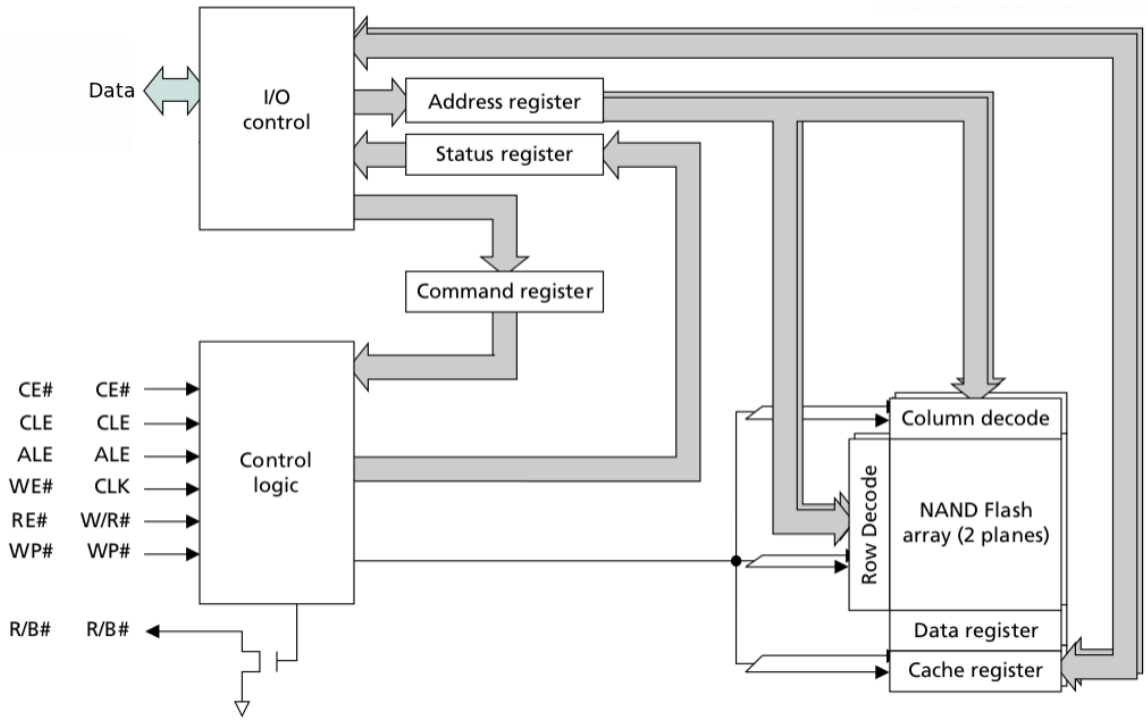

Figure 3.4: LUN Functional Block Diagram

3.4 NAND Flash Commands

- READ PAGE
- PROGRAM PAGE
- BLOCK ERASE
- READ ID
- READ STATUS
- RESET

3.4.1 NAND Flash Caching Mode Commands

During normal page operations, the data and cache registers act as a single register. During cache operations, the data and cache registers operate independently to increase data throughput.

READ PAGE CACHE MODE

The PAGE READ CACHE MODE command enables the user to pipeline the next sequential access from the array while outputting the previously accessed data. Data is initially transferred from the NAND Flash array to the data register. If the cache register is available (not busy), the data is quickly moved from the data register to the cache register as shown in Figure 3.5. After the data has been transferred to the cache register, the data register is available and can start to load the next sequential page from the NAND Flash array[1].

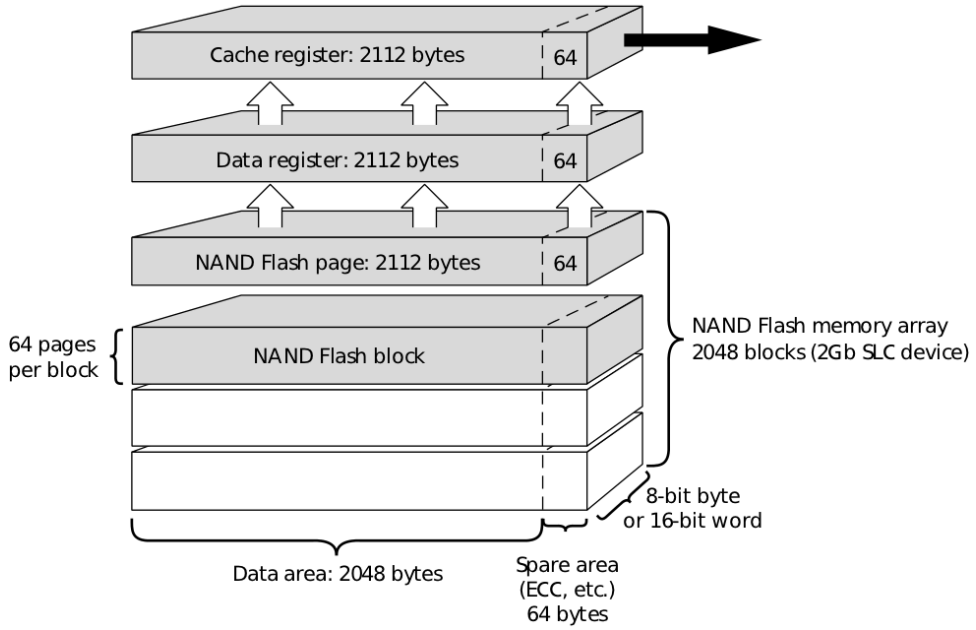

Figure 3.5: Read Page Cache Sequential Architecture

Using the PAGE READ CACHE MODE command delivers a 33% performance improvement over a traditional PAGE READ command on an 8-bit I/O device, with throughput up to 31 MB/s. On 16-bit I/O devices, throughput can be increased to 37 MB/s, delivering as much as a 40% performance improvement over normal PAGE READ operations.

PROGRAM PAGE CACHE MODE

PROGRAM PAGE CACHE MODE provides a performance improvement over normal PROGRAM PAGE operations. PROGRAM PAGE CACHE MODE is a double-buffered technique that enables the controller to input data directly to the cache register and uses the data register as a holding area to supply data for programming the array(see Figure 3.6). This frees the cache register so that the next sequential page operation can be loaded in parallel[1].

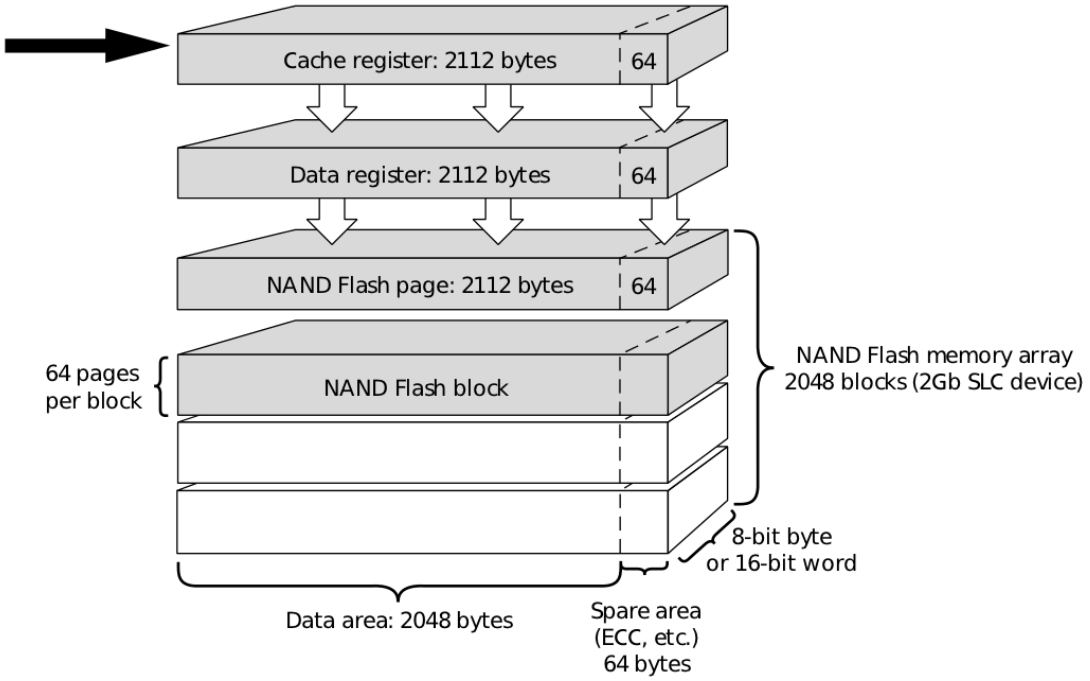

Figure 3.6: Program Page Cache Architecture

As with the PAGE READ CACHE MODE command, the data register is used to maintain the data throughput during the entire programming cycle. This frees the cache register to receive the next page of data from the controller.

CHAPTER 4

NAND Flash Controller

4.1 Organization of NAND Flash Controller

4.1.1 Registers

ID register : Stores the information about the NAND flash device.

Command register : Stores the commands the NAND flash controller is to execute.

Address register : Stores the address the controller is to *read from* or *write to* or *erase*.

Status registers : Stores the status of NAND flash device[6].

4.1.2 Control Logic

The main control logic controls the signals going to the NAND flash device via ONFi interface. By setting appropriate values to the registers mapped to signals like CE, CLE, ALE, WE and RE, it can perform different operations on NAND flash.

4.1.3 Buffers

The NAND Flash controller includes two buffers so the host does not have to wait for the controller to finish an operation to use a buffer. While the controller uses one buffer, the host has control of the other buffer.

Register Address	READ/WRITE	Name	Register Function	
0xFF0	Read	ID register 0	The registers the host should read following a READ ID operation.	
0xFF1	Read	ID register 1		
0xFF2	Read	ID register 2		
0xFF3	Read	ID register 3		
0xFF4	R/W	Block address [7:0]	Used to address the blocks and pages of the NAND Flash.	
0xFF5	R/W	Block address [15:8]		
0xFF6	R/W	Block address [24:16]		
0xFF7	Read	Part type and ECC	Bit [0]	0 = x8 device 1 = x16 device
			Bit [2:1]	00 = 2Gb 01 = 4Gb 10 = 8Gb
			Bit [6]	0 = No ECC 1 = ECC module connected
0xFF8	R/W	Buffer number	Bit [0]	0 = Host controls Buffer 1 1 = Host controls Buffer 2
0xFF9	Read	Status register	Used for reading the status from the NAND Flash.	
0xFFA	Write	Command register	Holds the commands the controller is to execute. 00h = PAGE READ 60h = BLOCK ERASE 70h = READ STATUS 80h = PROGRAM PAGE 90h = READ ID FFh = RESET (NAND Flash) Also holds the final commands of certain operations. 10h = PROGRAM PAGE 30h = PAGE READ D0h = BLOCK ERASE	

Figure 4.1: Register Description

4.2 NFC Functional Description

NAND Flash Controller has two interfaces. The Target interface is the ONFi interface. The Host interface is designed to be compliant with NVM express as well as any host processor. Figure 4.2 shows the interface signals coming from/going to the Host via Host interface and coming from/going to the NAND flash device via ONFi interface[7].

NVM express sends out 32-bit data and hence, the data bus on the Host side is 32-bit wide while NAND Flash supports either 8-bit or 16-bit wide data bus. So, the data bus on the Target side can be either 8-bit or 16-bit wide depending on the device. The host sends out commands on the data bus and addresses on the address bus. According to the command, the NAND Flash Controller takes certain action like read to or write from a certain location on the NAND flash device specified by the address from the host. The data management in the controller is managed by the two buffers.

4.2.1 NAND Flash Controller Block Diagram

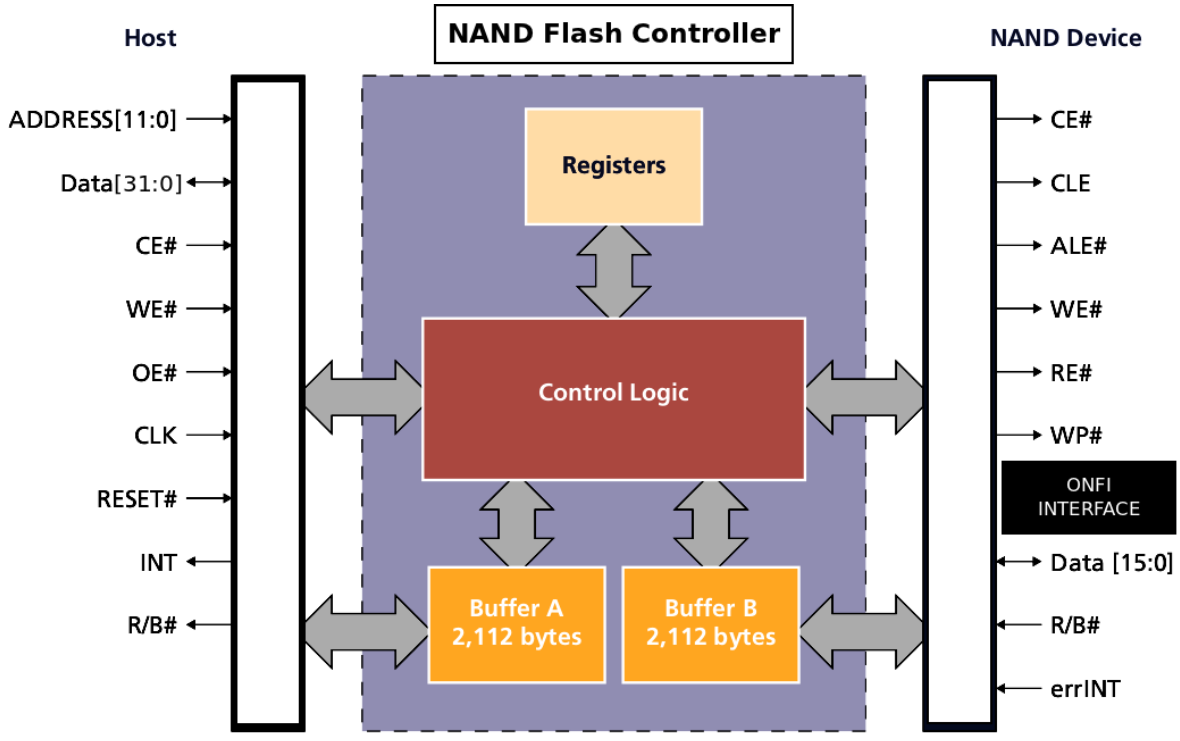

Figure 4.2: NAND Flash Controller Block Diagram

4.3 Interface Signal Descriptions

4.3.1 Host Interface

Signals which begin with 'wr_' are *wires*.

Signals which begin with 'rg_' are *registers*.

Signals which end with '_l' are *active low* signals.

The width of the signal is defined within '[]'.

The *type* of the signal indicates whether the signal is an input/output to the NFC.

Table 4.1: Host Interface Signals

Signal	*Type*	*Description*
wr_address_from_nvm[11:0]	input	address input to the NFC from the host
wr_data_from_nvm[31:0]	input	data input to the NFC from the host
rg_data_to_nvm[31:0]	output	data output to the host from the NFC
wr_nand_ce_l	input (active low)	chip enable signal from the host, for NFC whether to accept input or not
wr_nand_we_l	input (active low)	write enable allows data to be written by the host in the current buffer of NFC
wr_nand_oe_l	input (active low)	output enable allows data to be read by the host from NFC
wr_nand_reset_l	input (active low)	resets the NFC control logic
rg_interrupt	output	interrupt to host
rg_ready_busy_l	output (active low)	ready/busy status of NFC to the host

4.3.2 Target(ONFi) Interface

The NAND Flash Controller module sends/receives signals to/from another module related to the NAND Flash Device, namely *NAND Flash Target* via the ONFi interface. The ONFi interface, hence has two divisions of signals - Signals sent to/received from NAND Flash Controller Module and Signals sent to/received from NAND Flash Target Module.

A signal which is a *wire* for one module, will be interconnected to a signal which is a *register* for the other module and vice-versa. They are interconnected by a module, namely *InterConnectionModule* in Bluespec. The naming of the signals in this interface is similar to that of the Host interface.

Table 4.2: Target(ONFi) Interface Signals for NAND Flash Controller

Signal	*Type*	*Description*
rg_data_to_flash[15:0]	output	data output to the target from the NFC
wr_data_from_flash[15:0]	input	data input from the target to the NFC
rg_onfi_ce_l	output (active low)	chip enable to accept input from NFC
rg_onfi_re_l	output (active low)	read enable allows data to be read from NAND flash
rg_onfi_we_l	output (active low)	write enable allows data to be written on NAND flash
rg_onfi_wp_l	output (active low)	write protect to prevent writing data to NAND flash
rg_onfi_cle_l	output (active low)	command latch enable provides command input to the target via data bus
rg_onfi_ale_l	output (active low)	address latch enable provides address input to the target via data bus
wr_interrupt	input	interrupt from target
wr_ready_busy_l	input (active low)	ready/busy status of the NAND flash to the NFC

Table 4.3: Target(ONFi) Interface Signals for NAND Flash Target

Signal	*Type*	*Description*
rg_data_to_nfc[15:0]	output	data output to the NFC from the Target
wr_data_from_nfc[15:0]	input	data input from the NFC to the Target
wr_onfi_ce_l	input (active low)	chip enable to accept input from NFC
wr_onfi_re_l	input (active low)	read enable allows data to be read from NAND flash
wr_onfi_we_l	input (active low)	write enable allows data to be written on NAND flash
wr_onfi_wp_l	input (active low)	write protect to prevent writing data to NAND flash
wr_onfi_cle_l	input (active low)	command latch enable provides command input to the target via data bus
wr_onfi_ale_l	input (active low)	address latch enable provides address input to the target via data bus
rg_t_interrupt	output	interrupt to NFC
rg_t_ready_busy_l	output (active low)	ready/busy status of the NAND flash to the NFC

4.4 NAND Flash Command Sequences

4.4.1 READ PAGE

READ PAGE is initiated when the NAND Flash Controller receives the command '00h' from the Host. The NFC realizes that the data sent is a command by the address sent by the Host, which is the address of the *command register*. The Host sends the second and final command of *Read Page* i.e., '30h' after loading the address of the page(to be read) onto the *address register*. Then, the NFC goes from idle state to command input state as shown in figure 4.3 and sends the command to the NAND Flash Target.

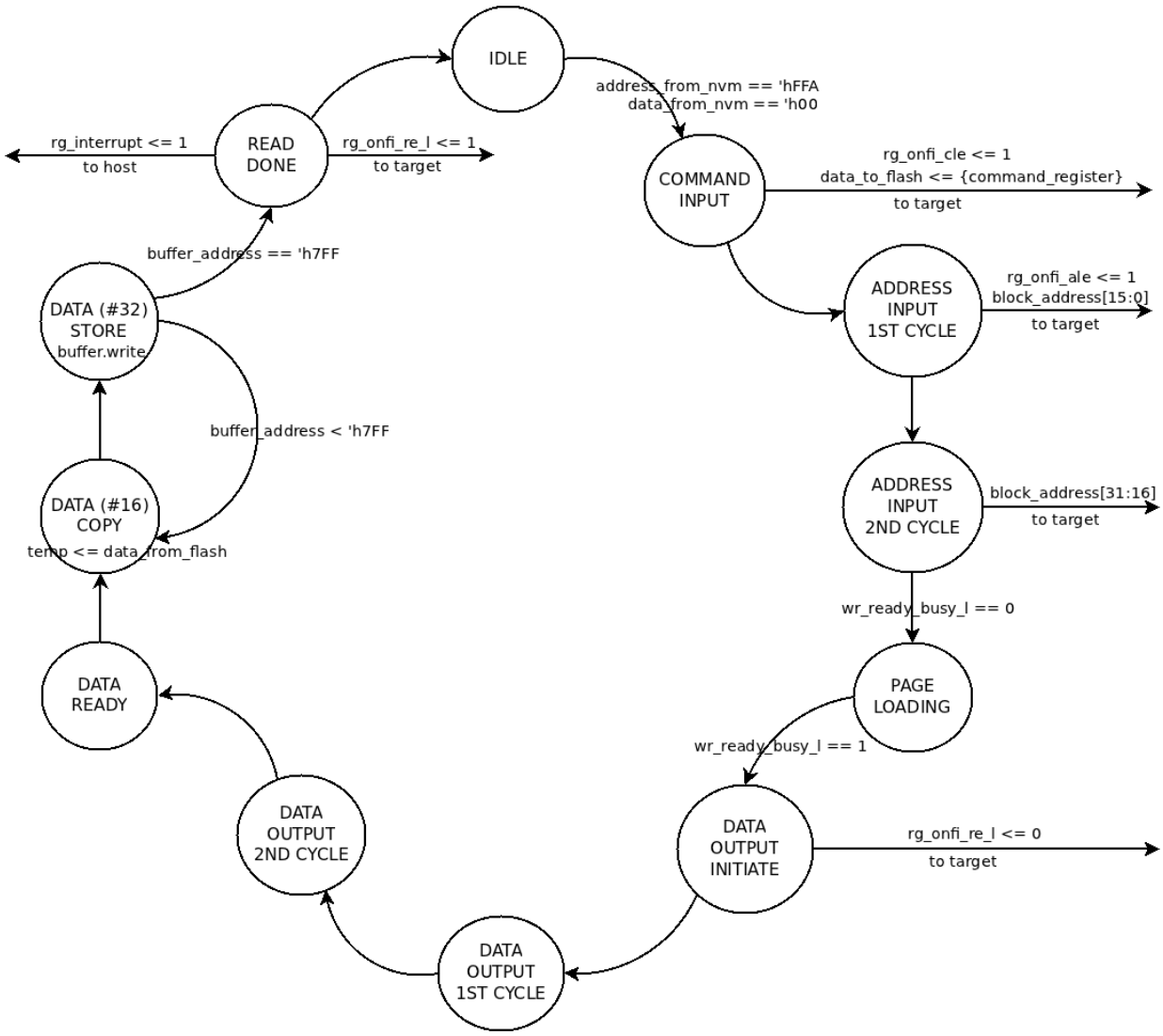

Figure 4.3: Read Page State Machine

After sending the command, the NAND Flash Controller initiates the *address input* cycle and sends the address of the page to be read to the NAND Flash Target. Since the data bus width is 16-bit, it takes two cycles to send the address. The address contains the block address and the page address. In case of multiple LUNs, the LUN address is also included in the address sent by the Host.

The NAND Flash Target makes the ready/busy signal go low indicating that it is busy copying the page from the address location to the data register of the Target, from which

the NFC can read the data. After the copying is done, the Target makes the ready/busy signal go high. Now, the NFC initiates the *data output* cycle and starts copying the data from the Target onto the data buffer(s).

The minimum width of the data stored in buffer(s) per cycle is 32-bit but the data bus width of ONFi interface is 16-bit. So, in a two-cycle process, the first- cycle's 16-bit data is stored in a temporary register which is copied along with the next-cycle's 16-bit data onto the buffer. This happens for a number of cycles until the last entity of data is read from the Target and copied onto the buffer(s). The *buffer address* indicates the address of the current location of data in the data buffer(s). So, it is incremented every time a 32-bit data is copied onto the buffer.

The NAND Flash Controller sends an interrupt to the Host indicating that the *Read Page* process is complete, terminates the data output cycle and goes back to idle state. It remains in idle state until the next command is received.

The READ PAGE command uses only the data register of NAND Flash Device as an intermediate storage in the process of data transfer. There are other kinds of READ PAGE commands which make use of the *cache register* in NAND Flash Device.

- READ PAGE CACHE SEQUENTIAL
- READ PAGE CACHE RANDOM
- READ PAGE CACHE LAST
- READ PAGE MULTI-PLANE

4.4.2 PROGRAM PAGE

PROGRAM PAGE is initiated when the NAND Flash Controller receives the command '80h' from the Host. The NAND Flash Controller realizes that the data sent is a *com-*

mand by the address sent by the Host, which is the address of the *command register*. The Host sends the second and final command of PROGRAM PAGE i.e., '10h' after loading the address(of the location where the page is to be written) onto the *address register* and storing the page data in the data buffer(s). Then, the NAND Flash Controller goes from idle state to command input state as shown in figure 4.4 and after initiating the command input cycle, the NFC sends the command to the NAND Flash Target.

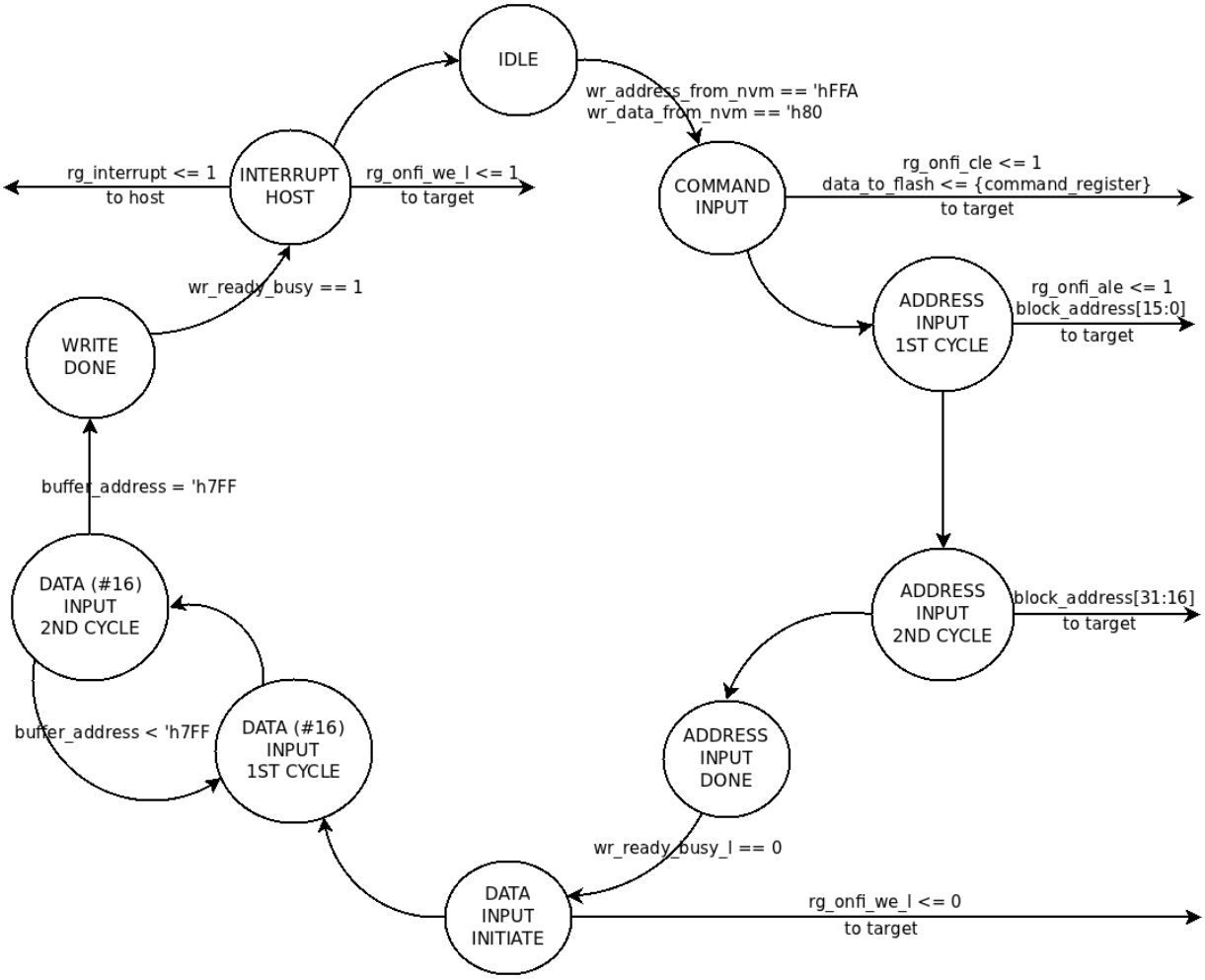

Figure 4.4: Program Page State Machine

After sending the command, the NAND Flash Controller initiates the *address input* cycle and sends the address of the location (where the page is to be written) to the NAND Flash Target. Since the data bus width is 16-bit, it takes two cycles to send the address.

The address contains the block address and the page address. In case of multiple LUNs, the LUN address is also included in the address sent by the Host.

The NAND Flash Target makes the ready/busy signal go low indicating that it is done copying the address and is about to start copying the data from the buffers onto the data register and finally, onto the page location(referred by the address) on the NAND Flash Device. Now, the NFC initiates the *data input* cycle by making the write enable(active low) signal go low and starts copying the data from the data buffer(s) to the NAND Flash Target.

The minimum width of the data coming out of buffer(s) per cycle is 32-bit but the data bus width of ONFi interface is 16-bit. So, the 32-bit data takes two cycles to get written on NAND Flash Device. The *buffer address* indicates the address of the current location of data in the data buffer(s). So, it is incremented every time a 32-bit data is copied onto the buffer i.e, every two cycles.

After the writing of data is done, the NAND Flash Target makes the ready/busy signal go high indicating that the write process is done and the NAND Flash Target is free now. The NAND Flash Controller sends an interrupt to the Host indicating that the PROGRAM PAGE process is complete, terminates the data input cycle and goes back to idle state. It remains in idle state until the next command is received.

The PROGRAM PAGE command uses only the data register of NAND Flash Device as an intermediate storage in the process of data transfer. There are other kinds of PROGRAM PAGE commands which make use of the *cache register* in NAND Flash Device.

- PROGRAM PAGE CACHE
- PROGRAM PAGE MULTI-PLANE

4.4.3 BLOCK ERASE

Read/Write operations in a NAND Flash Device are page-based while Erase operation is block-based. A whole block containing a set of pages can be erased at a time. This is one of the advantages of NAND Flash. The block-erase process is illustrated as a state diagram in Figure 4.5.

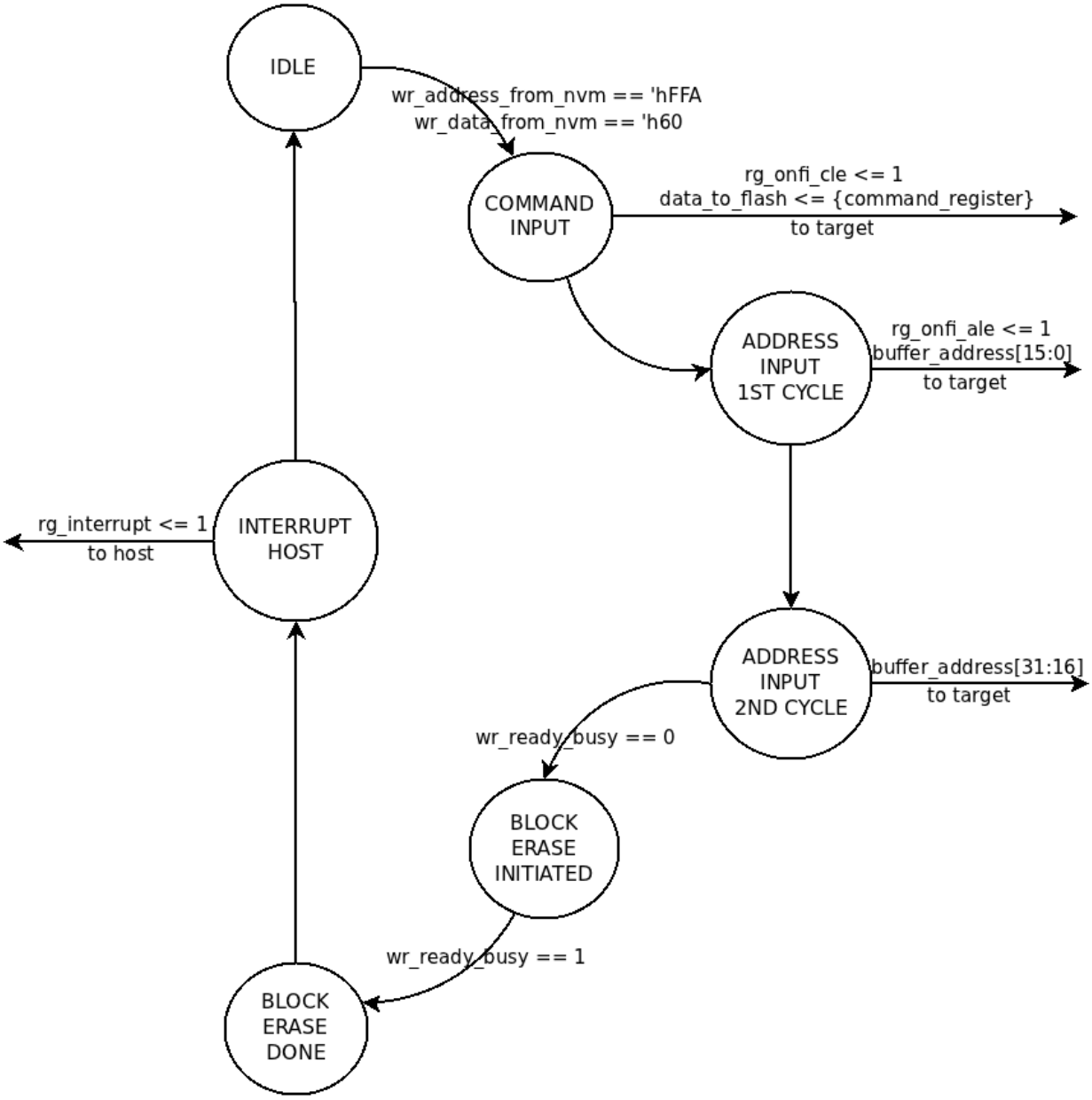

Figure 4.5: Block Erase State Machine

BLOCK ERASE is initiated when the NAND Flash Controller receives the command '60h' from the Host. The NAND Flash Controller realizes that the data sent is a *command* by the address sent by the Host, which is the address of the *command register*. The Host sends the block-address via data bus and the sends the location of its storage(*address register*) via address bus.

The Host sends the second and final command of BLOCK ERASE i.e., 'D0h' after the address(of the block to be erased) is loaded onto the *address register*. The NAND Flash Controller goes from idle state to command input state as shown in figure 4.5 and after initiating the command input cycle by making the command latch enable signal go high, the NAND Flash Controller sends the command to the NAND Flash Target.

After sending the command, the NAND Flash Controller initiates the *address input* cycle and sends the address of the block (to be erased) to the NAND Flash Target. Since the data bus width is 16-bit, it takes two cycles to send the address. In case of multiple LUNs, the LUN address is also included in the address sent by the Host.

The NAND Flash Target makes the ready/busy signal go low indicating that it is done copying the address and the erase operation is under way. After the erase operation is done, the NAND Flash Target makes the ready/busy signal go high indicating that the block is erased and the NAND Flash Target is not busy anymore.

The NAND Flash Controller sends an interrupt to the Host indicating that the BLOCK ERASE process is complete and goes back to idle state. It remains in idle state until the next command is received.

4.4.4 READ ID

READ ID is initiated when the NAND Flash Controller receives the command '90h' from the Host. The NAND Flash Controller realizes that the data sent is a *command* by the address sent by the Host, which is the address of the *command register*. READ ID has no second command unlike the operations discussed earlier. There is no need for an address input too.

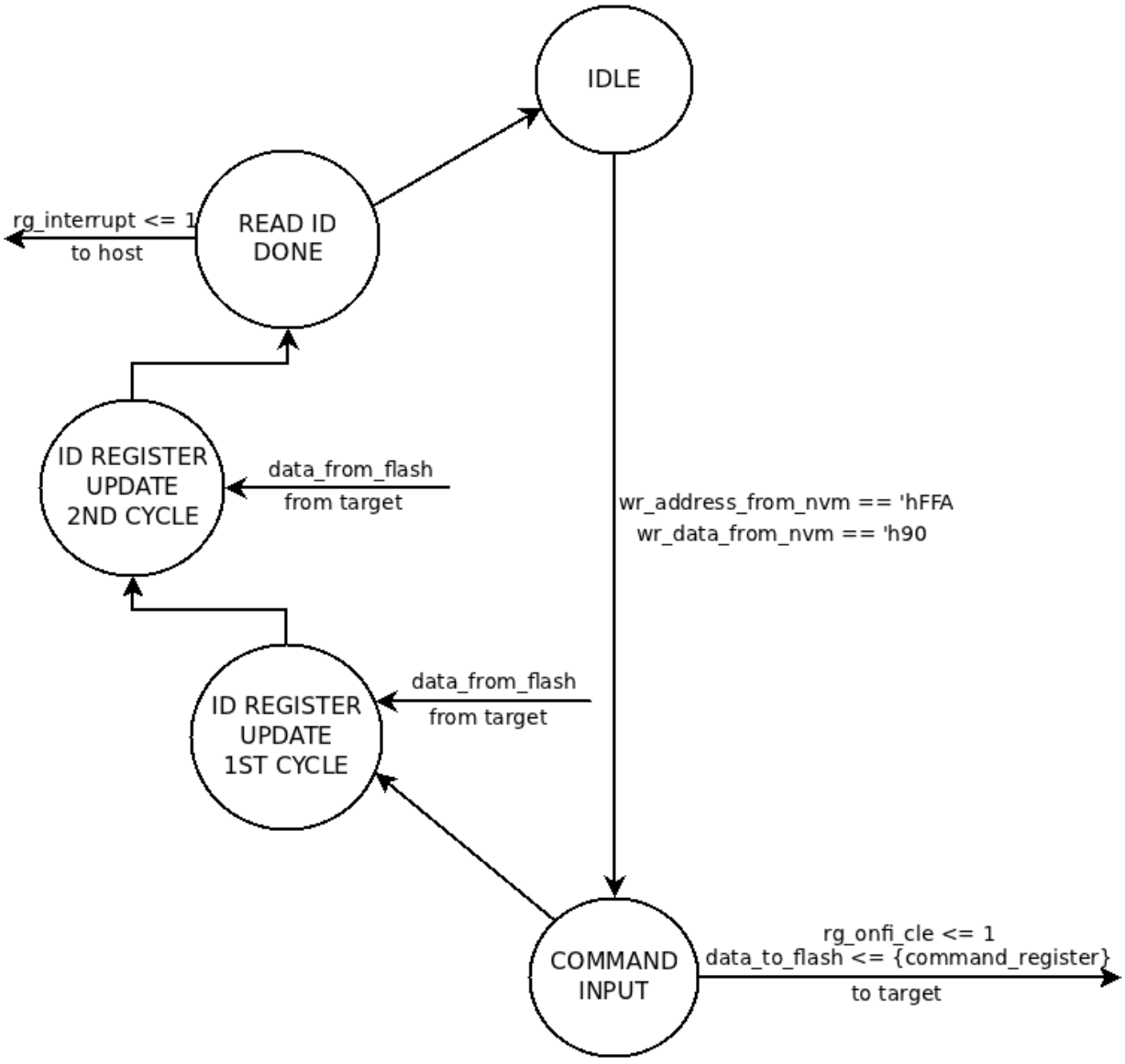

Figure 4.6: Read ID State Machine

The NAND Flash Controller goes from idle state to command input state as shown in figure 4.6 and after initiating the command input cycle by making the command latch enable signal go high, the NAND Flash Controller sends the command to the NAND Flash Target.

After receiving the command, the NAND Flash Target sends the ID data to the NAND Flash Controller. The NAND Flash Controller stores the received data in the *ID register*. Since the data bus width of ONFi interface is 16-bit and the ID data width is 32-bit, it takes two cycles for the NAND Flash Target to send the data to the NAND Flash Controller.

The NAND Flash Controller sends an interrupt to the Host indicating that the READ ID process is complete and goes back to idle state. It remains in idle state until the next command is received.

The READ ID (90h) command is used to read identifier codes programmed into the target. There is a different READ ID command called READ UNIQUE ID (EDh) which is used to read a unique identifier programmed into the target. This command is accepted by the target only when all die (LUNs) on the target are idle. Writing 'EDh' to the command register puts the target in read unique ID mode. The target stays in this mode until another valid command is issued.

Another kind of READ ID command is READ PARAMETER PAGE command. The READ PARAMETER PAGE (ECh) command is used to read the ONFI parameter page programmed into the target. This command is accepted by the target only when all die (LUNs) on the target are idle. Writing 'ECh' to the command register puts the target in read parameter page mode. The target stays in this mode until another valid command is issued.

ID data is generally of 4-byte size. There can be an additional byte of data depending on the device. The figure 4.7 shows the typical Read ID response of a Micron NAND Flash Device[5].

	Option	I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	I/O0	Value[1]
Byte 0 – Manufacturer ID										
Manufacturer	Micron	0	0	1	0	1	1	0	0	2Ch
Byte 1 – Device ID										
MT29F2G08AAD	2Gb, x8, 3V	1	1	0	1	1	0	1	0	DAh
MT29F2G16AAD	2Gb, x8, 3V	1	1	0	0	1	0	1	0	CAh
MT29F2G08ABD	2Gb, x8, 1.8V	1	0	1	0	1	0	1	0	AAh
MT29F2G16ABD	2Gb, x16, 1.8V	1	0	1	1	1	0	1	0	BAh
Byte 2										
Number of die per CE#	1							0	0	00b
Cell type	SLC					0	0			00b
Number of simultaneously programmed pages	1			0	0					01b
Interleaved operations between multiple die	Not supported		0							0b
Cache programming	Supported	1								1b
Byte value	MT29F2Gxxxxx	1	0	0	0	0	0	0	0	80h
Byte 3										
Page size	2KB							0	1	01b
Spare area size (bytes)	64B						1			1b
Block size (w/o spare)	128KB			0	1					01b
Organization	x8		0							0b
	x16		1							1b
Serial access (MIN)	25ns	1				0				1xxxb
Serial access (MIN)	35ns	0				0				0xxx0b
Byte value	MT29F2G08AAD	1	0	0	1	0	1	0	1	95h
	MT29F2G16AAD	1	1	0	1	0	1	0	1	D5h
Byte value	MT29F2G08ABD	0	0	0	1	0	1	0	1	15h
	MT29F2G16ABD	0	1	0	1	0	1	0	1	55h
Byte 4										
Reserved								0	0	00b
Planes per CE#	1					0	0			00b
Plane size	2Gb		1	0	1					101b
Reserved		0								0b
Byte value	MT29F2Gxx	0	1	0	1	0	0	0	0	50h

Notes: 1. b = binary; h = hexadecimal

Figure 4.7: Read ID Response of a Micron NAND Flash Device

4.4.5 READ STATUS

READ STATUS is initiated when the NAND Flash Controller receives the command '70h' from the Host. The NAND Flash Controller realizes that the data sent is a *command* by the address sent by the Host, which is the address of the *command register*. READ STATUS has no second command and there is no need for address input too.

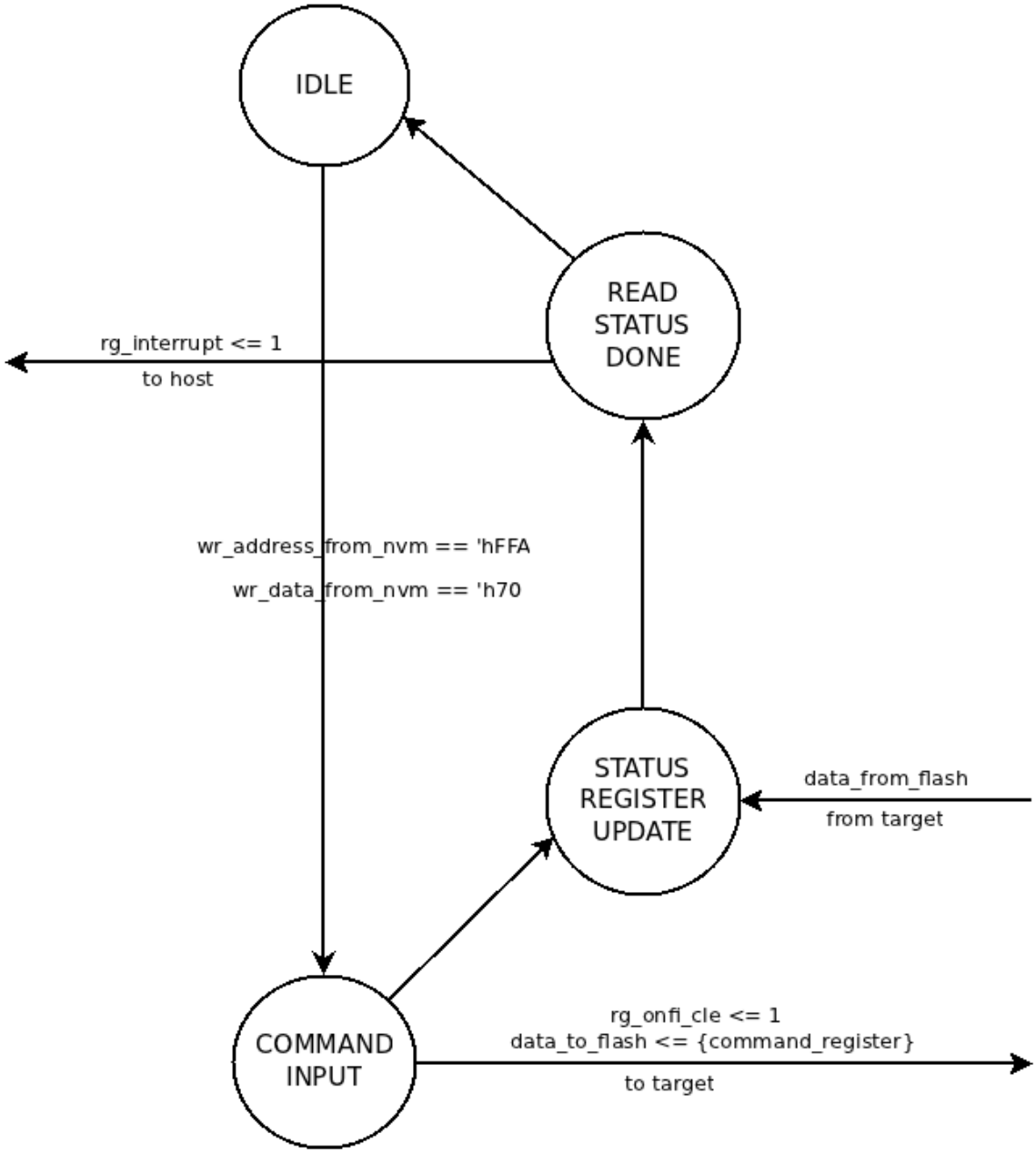

Figure 4.8: Read Status State Machine

The NAND Flash Controller goes from idle state to command input state as shown in figure 4.8 and after initiating the command input cycle by making the command latch enable signal go high, the NAND Flash Controller sends the command to the NAND Flash Target.

After receiving the command, the NAND Flash Target sends the status of the last-selected LUN to the the NAND Flash Controller. This command is generally accepted by the NAND Flash Target even when it is busy. Status data is of 8 bits i.e., 1 byte and so, it is stored in an 8-bit *status register* by the NAND Flash Controller.

The NAND Flash Controller sends an interrupt to the Host indicating that the BLOCK ERASE process is complete and goes back to idle state. It remains in idle state until the next command is received.

To know the status of a particular LUN, READ STATUS ENHANCED command is used. The READ STATUS ENHANCED (78h) command returns the status of the addressed die (LUN) on a target even when it is busy. This command is accepted by all die (LUNs), even when they are BUSY. Writing 78h to the command register, followed by the LUN address, puts the selected die (LUN) into read status mode. The selected die (LUN) stays in this mode until another valid command is issued. Die (LUNs) that are not addressed are deselected to avoid bus contention.

Use of the READ STATUS ENHANCED (78h) command is prohibited during the power-on RESET (FFh) command. It is also prohibited following some of the other reset, identification, and configuration operations.

4.4.6 RESET

RESET is initiated when the NAND Flash Controller receives the command 'FFh' from the Host. The NAND Flash Controller realizes that the data sent is a *command* by the address sent by the Host, which is the address of the *command register*. RESET has no second command and there is no need for address input too.

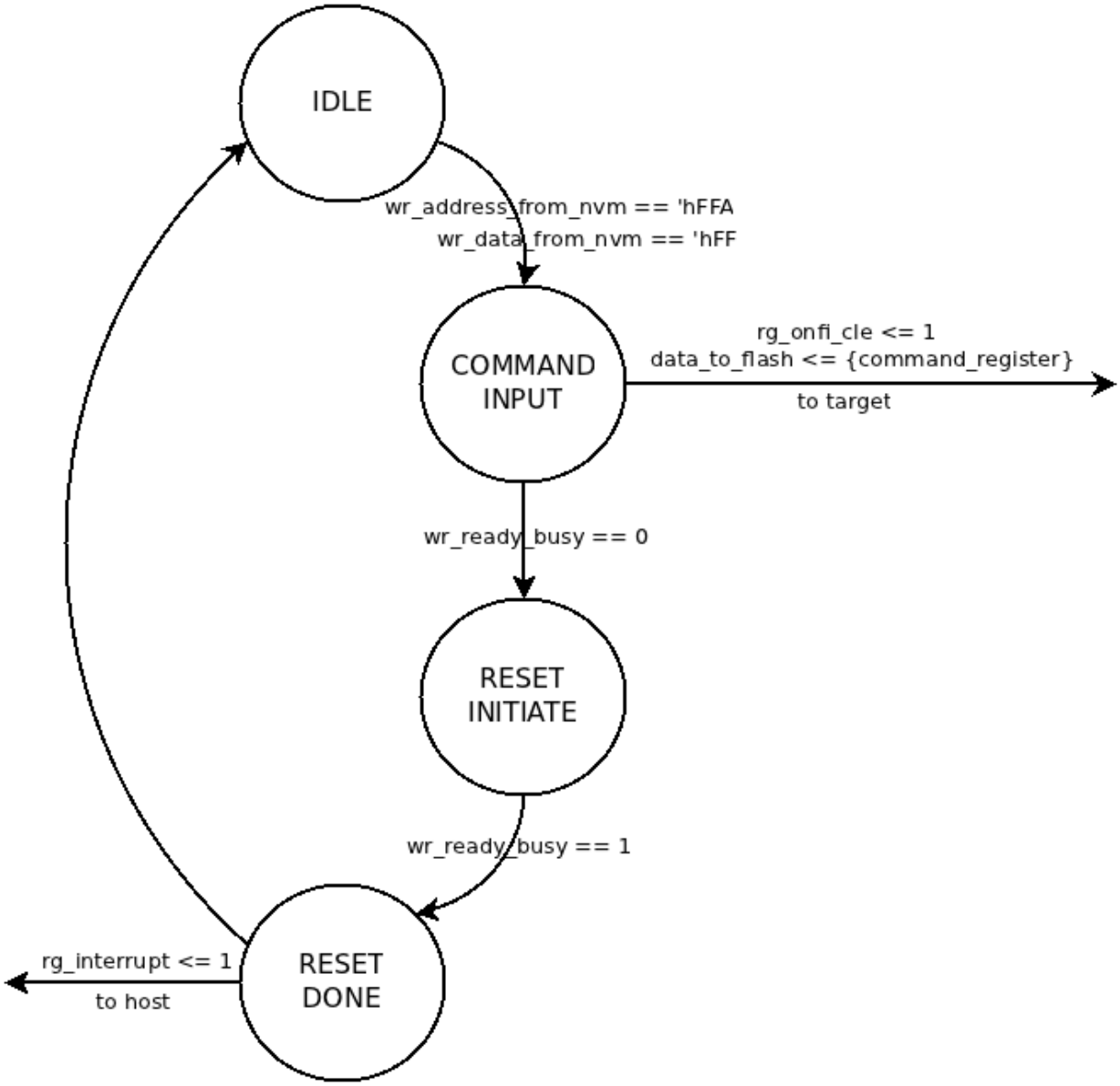

Figure 4.9: Reset State Machine

The NAND Flash Controller goes from idle state to command input state as shown in figure 4.9 and after initiating the command input cycle by making the command latch enable signal go high, the NAND Flash Controller sends the command to the Target.

The NAND Flash Target makes the ready/busy signal go low indicating that the reset operation is under way. After the reset operation is done, the NAND Flash Target makes the ready/busy signal go high. This indicates to the NAND Flash Controller that the Target is free which means that the reset operation is done. The NAND Flash Controller sends an interrupt to the Host indicating that the RESET process is complete and goes back to idle state. It remains in idle state until the next command is received.

If this command is issued while a PROGRAM or ERASE operation is occurring on one or more die (LUNs), the data may be partially programmed or erased and is invalid. The command register is cleared and ready for the next command. The data register and cache register contents are invalid.

RESET must be issued as the first command to each target following power-up. Use of the READ STATUS ENHANCED (78h) command is prohibited during the power-on RESET. To determine when the target is ready, READ STATUS (70h) is used.

4.5 Verification

4.5.1 NAND Flash Controller Verification Setup

The design under verification is the NAND Flash Controller module, which is connected to the NAND Flash Target module via ONFi interface as shown in figure 4.10.

The test case commands are provided as input to the DUV module(NAND Flash Con-

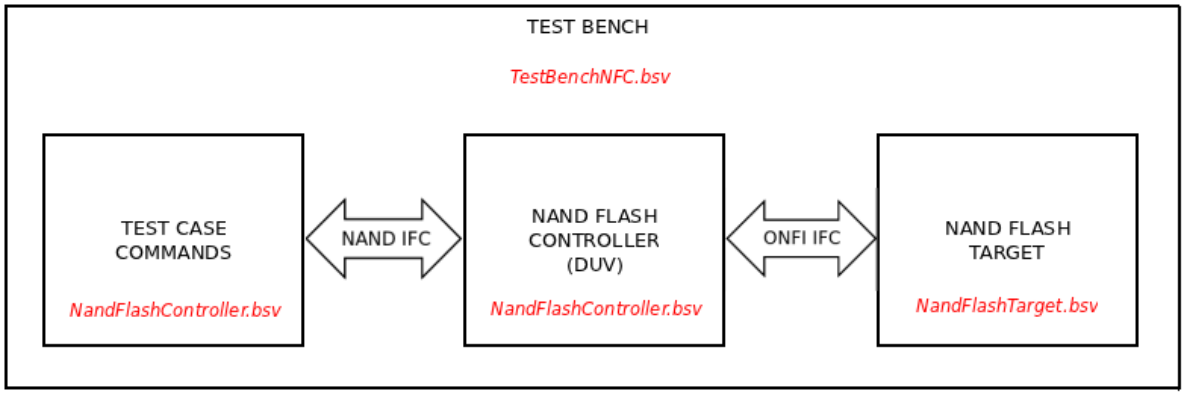

Figure 4.10: Verification Setup for verifying NAND Flash Controller

troller module) which initiates the communication between the NAND Flash Controller module and the NAND Flash Target module. Process-specific operations are carried out by the two modules and after the process is complete, the results are verified.

4.5.2 NAND Flash Controller Verification Test Cases

Read Page

Address of the page to be read is given by the test bench and it is verified whether correct data is read.

Program Page

Address and data are provided by the test bench and whether the correct data is written on the target in the right address location is verified.

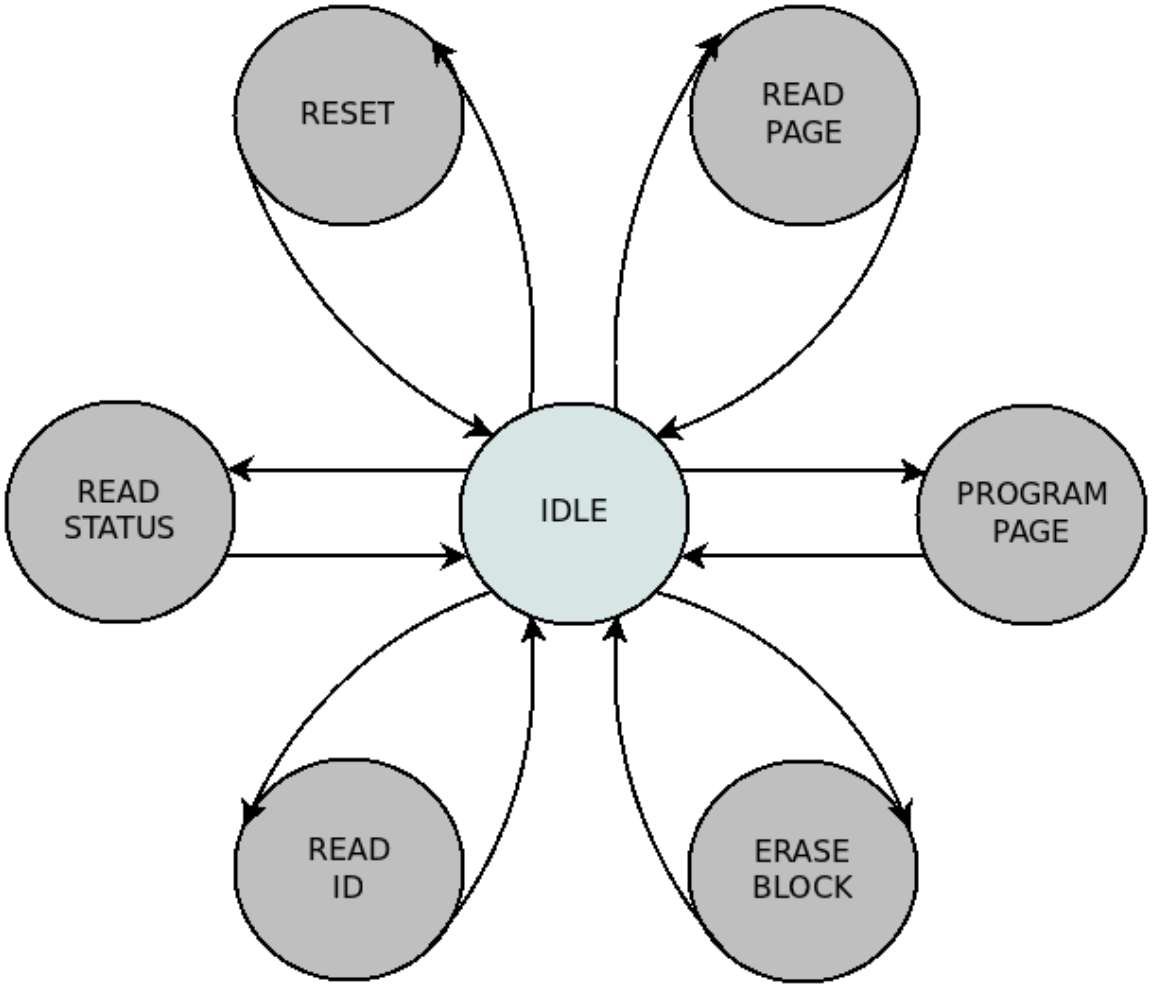

Figure 4.11: NAND Flash Controller Test Cases

Block Erase

Address of the block to be erased is provided by the test bench and later, checked whether the block is cleared or not.

Read ID

The READ ID command is given by the test bench and the data received is verified whether it matches with the device ID.

Read Status

The test bench provides data and address to the NFC such that it operates in a known state. Then, READ STATUS command is given by the test bench and the data received is verified.

Reset

RESET command is given by the test bench to the NFC and later, verified whether the NAND flash is reset.

4.5.3 NVM Express-driven NFC Verification Setup

The design under verification is the NAND Flash Controller module which is driven by NVM express Controller module. The NAND Flash Controller module is connected to the NAND Flash Target module on the other side via ONFi interface as shown in figure 4.10.

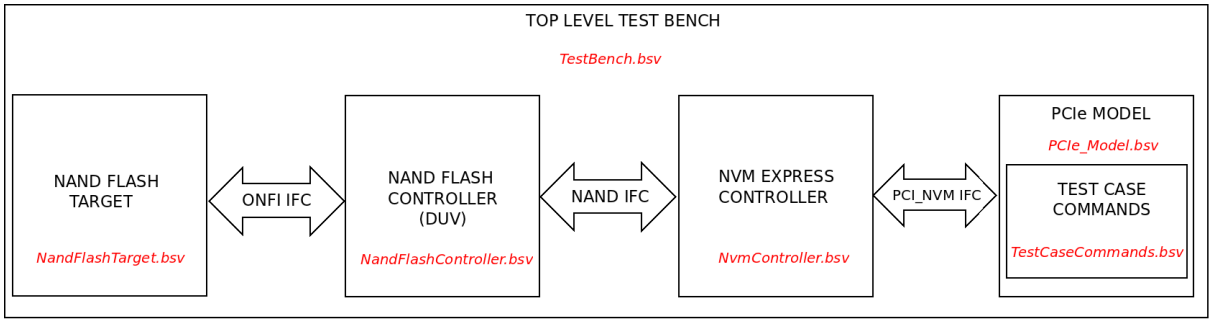

Figure 4.12: Verification Setup for verifying NVMe-driven NAND Flash Controller

The test case commands are provided by the PCIe model in this setup. Process-specific operations are carried out by the other modules and after the process is complete, the results are verified.

4.5.4 NVM Express-driven NFC Verification Test Cases

NVM express provides test cases for the following operations :

- Read Page
- Program Page
- Read ID
- Read Status
- Reset

NVM express does not provide test case for BLOCK ERASE operation.

CHAPTER 5

Conclusion and Future Work

The performance and power improvements over the latest generation of ONFi are definitely game changing. New SSD price points and improvements in system performance are inevitable. In an industry that lacks standardization in commands, timing, architecture, and even pin-out, NAND Flash is still being rapidly adopted for solid-state drives (SSDs), mobile drive applications, servers, and other computing and consumer applications. These advantages can be wasted in the implementation, however, especially given the software and hardware architectural variations used today. Designers should pay attention to the following:

- I/O selection (vendor flexibility, power improvements, data reliability)
- Device interleaving (performance, channel efficiency, latency)
- Command pipeline (channel efficiency, latency, command efficiency)
- ECC non-blocking (improves channel-to-channel latency)
- Partial page operation

The increase in page size and write performance is a growing trend as NAND process technology shrinks, and it works very nicely with the increase in bus speed provided by ONFi 3. Program erase cycle times are also increasing, which means devices are not accessible for longer periods of time, although the problem can be mitigated by more channels and fewer devices or by system software. Next-generation products using ONFi 3 will provide an improved user experience at a better price point than at any other time in the history of NAND Flash.

APPENDIX A

Additional Commands

Future work can be done to improve the present design of NAND Flash Controller like exploiting CACHE MODE commands much more to improve the performance of NFC.

Adding a READ STATUS ENHANCED command can be useful to learn more information about a particular die. READ PAGE CACHE and PROGRAM PAGE CACHE commands can be used to speed-up the read and write processes.

COPYBACK operations make it possible to transfer data within a plane from one page to another using the cache register. The COPYBACK operation is a two-step process consisting of a COPYBACK READ (00h-35h) and a COPYBACK PROGRAM (85h-10h) command. This is particularly useful for block management and wear leveling.

New commands keep popping-up every once in a while which can be used appropriately to improve the performance of NAND Flash operations. So, keeping up-to-data with this evolving command-chain is important.

APPENDIX B

NAND Flash Controller with AXI interface

AXI interface is a standard interface and hence, the host processor can directly communicate with the NAND Flash Controller when connected via AXI interface. NVM Express controller can also communicate with the NAND Flash controller via AXI interface.

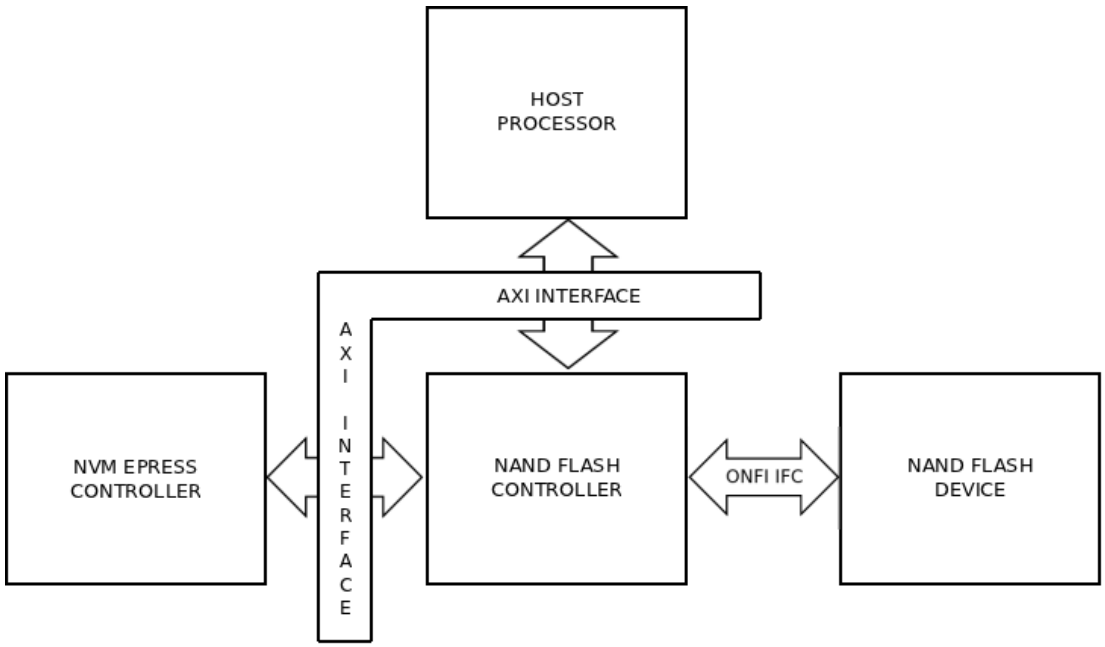

Figure B.1: NAND Flash Controller with AXI interface

REFERENCES

[1] Micron, *An Introduction to NAND Flash*, revision b ed., April 2010.

[2] R. S. Nikhil and K. Czeck, *BSV by Example*. Bluespec, Inc, 2010.

[3] Bluespec, Inc, *Bluespec System Verilog Reference Guide*, revision: 17 ed., 2012.

[4] ONFi Workgroup, *Open NAND Flash Interface Specification*, revision 3.0 ed., 9 March 2011.

[5] Micron, *NAND Flash Memory*, revision e ed., March 2010.

[6] Xilinx, *TN-29-06: NAND Flash Controller*, revision b ed., June 2007.

[7] L. C. Rino Micheloni and A. Marelli, *Inside NAND Flash Memories*. Springer, 2010.