Overview of the NAND Flash High-Speed Interfacing and Controller Architecture

Nina Mitiukhina

Abstract—The paper is dedicated to NAND Flash array architecture and its interfacing with NAND Flash Controller. Inherent properties of NAND Flash non-volatile memory and the demand of high-speed high performance NAND interfacing makes NAND Flash Controller design a difficult challenge. After a brief introduction of the market tendencies, we provide background information on NAND Flash, following the device operation and System Controller functions. An example of controller operation is followed by the study of latest interface advancements according to ONFI 3.0 specifications. In the end, we present specific NAND Flash controller features such as ECC, Wear Leveling and Bad Block Management.

Index Terms—NAND Flash, DDR, toggle mode, asynchronous SDR clock generation, controller architecture, interfacing, ECC, wear leveling, bad block management.

I. INTRODUCTION

NOWADAYS NAND Flash memory pervades countless number of electronic applications, virtually all of them. It’s availability, low cost, in combination with high densities, makes up the choice in favor of NAND Flash on board of applications traditionally linked to other types of memory (such as EEPROM, NOR or HDD). In coming years, NAND Flash seems to be an appealing choice for most of mobile devices and many more new applications, requiring a storage media. We can consider memory environment has changed since 2001, when growth rate for NAND surpassed the “Moore law” for processor growth. In addition, MLC devices with 2,3 bit/cell soon entered volume production.

Fig. 1 Pricing comparison between NAND Flash, DRAM and HDD from 1990 till now.

Fig. 2 Performance demand with the growth of storage

This evolution in semiconductor industry aimed mainly to reduce the cost per MB of the memory itself in order to enable the new applications and open new markets.

All the way from 1990 up to 2012 there has been a distinctive price reduction for NAND Flash. Figure 1 clearly shows that HDD and NAND Flash pricing almost converge, starting from 2005. The change of price curve slope from 2005 to 2010 correlates with MLC introduction.

In contrast to the technology revolution, basic functionalities and performances of NAND Flash devices haven’t improved at the similar rate in the past decade[1].

Thus to face a growing industry demand for the high-speed NAND Flash Interface, continuous innovation and enhanced I/O speed need to be introduced to the system (Fig. 2).

For the forecast of the growth of the number of bits for NAND Flash, comparing to HDD, DRAM and NOR, please refer to Fig. 3.

II. NAND FLASH BACKGROUND INTRODUCTION

NAND Flash is a non-volatile memory, introduced by Toshiba and Samsung in 1989. Its benefits include high storage density, fast access time, low power requirements in operation and excellent shock resistance.

All of above listed merits are tempered by the inherent limitations of the technology. For example, due to production yield constraints, NAND flash ships with a number of bad blocks, that are not to be used. Unlike DRAM, NAND Flash requires a serialized data interface, similar to a hard disk, and so execute-in-place functionality can not be implemented. Moreover, with the time and after multiple erase cycles the memory will tend to “wear” and become less reliable. The bits stored in the chip can sometimes flip from 1 to 0 or vice versa[2].
NAND Flash SD memory vs. MMC

Raw NAND Flash device typically comes with the serial interface such as ONFI (one of the standards available). NAND Flash device can be found as discrete memory chips, or packaged with a controller to form SD memory card, Multi Media Card (MMC) or solid state disk (SSD). SSD and MMC cards are both widely used in consumer applications. SSD was designed to replace hard disk drive in computing systems. For embedded application, where memory storage other than hard disk-like device is required, designer has a choice of using raw NAND Flash device or SD or MMC memory. NAND Flash manufacturers offer single chip device that integrate both the MMC controller function with NAND Flash memory array composing eMMC (standard controlled by JEDEC). Figure 4 compares the system architecture of these two design approaches.

Giving the preference to raw NAND Flash chip, designer is responsible for all NAND Flash management tasks i.e. wear leveling, ECC and bad block management. If eMMC device is used, these management tasks are off-loaded to the eMMC device.

SLC vs MLC

NAND Flash memory usually uses one of the two memory technologies: Single Level Cell (SLC) or Multi Level Cell (MLC). As the name implies, 1 bit of information is stored in SLC and 2 or more bits of information can be stored in MLC. MLC offers higher memory density which results in lower cost per bit of storage. However, noise immunity and data integrity of MLC is a challenge, SLC performs better, because multiple levels of charge storage is written into each memory cell. SLC offers higher performance with faster page open and shorter programming time. SLC also demonstrates higher endurance, memory can be re-written hundreds times more than MLC devices. Typically, SLC offers write endurance of over 100,000 cycles but MLC is usually rated at 5,000 to 10,000 cycles.

To combat soft and hard errors in NAND Flash memory cells, error correction code (ECC) is required in both SLC and MLC devices. Due to lower data integrity, MLC device requires more sophisticated ECC. Please note that added complexity of ECC increases the gate count and power consumption of the controller.

Other than the differences in ECC requirement and write endurance cycle, SLC and MLC function in the same manner. Both topologies often employ the same external interface. Similarly, higher level design such as application software, device drivers and the controller hardware is independent.

NAND Flash Data Organization

The basic operation units for NAND Flash device is one page and one block of data. One page typically contains from 512 bytes to 16 kbytes. Each page also contains additional bits called the spare data or spare column. The spare data size may be as little as 16 bytes per page for page size of 512 bytes, or up to a thousand bytes per page for page size of 16 kbytes. The spare data is part of the page and is accessed at the same way as the main data page. The spare data can be used for marking bad block/page or for storing ECC parity bits.

Multiple pages are grouped together to form a block and multiple blocks are grouped together to form a plane. A Logical Unit (LUN) contains one or two planes. If the LUN contains 2 planes, interleaving read or write between the two plane is supported. A NAND flash die contains one or multiple LUN. Typical NAND Flash device may contain one or multiple NAND Flash dies.

Some commands such as read and program affect only one page while other commands such as erase affects the whole block. Commands such as reset affect the entire LUN. For a typical organization of a 8Gbit NAND device refer to Figure 5.

III. NAND FLASH DEVICE OPERATION

NAND Flash device makes use of 8 data signals and a few command signals for address, command, and data communication. Commands and address information are shifted into the device through the data pins during the command and address phases.
Read Operation

Two step process is required in order to read data from NAND Flash page. First, the page must be opened (as a result of the read command) for read, which usually takes up time range from 20 to 50 us, or the thousands of cycles for typical operating frequency. After a page is opened, all the data from the page is transferred internally to shift registers inside the NAND Flash, ready to be transferred to the external interface. Data can then be shifted out sequentially from the NAND device one byte at a time. Data transfer typically takes from 20 to 100 ns per byte for asynchronous device. Synchronous device are much faster with 5 to 25ns transfer time per byte. We will talk about this in detail in the later sections.

Data are read sequentially from within a page. To support random access, NAND Flash device allows user to change the read address within the page once the page is opened. However, the ECC requirement and the extra time needed for shifting the new address make random access very inefficient and rarely used.

Write Operation

To store data into NAND flash, a similar (but reversed) two step process is required. Data is first shifted into the shift registers of the NAND Flash device. After all the data of the page has shifted in, the user issues a program command to direct all the data from the data registers to be programmed into the specific page. Typical programming time is a few hundred microseconds per page for SLC and a few milliseconds for MLC. Variations such as two plane programming, cache programming and random programming are supported by many NAND Flash devices to reduce programming time.

Before new data can be written to a page, the page should be erased by the erase command, which sets all data bits to “1”.

During programming, either “1” or “0” can be written into each cell. After a “0” has been written to a cell, it cannot be changed back to “1” again with another write. Only the erase command can change the cell back to “1”. In other words, the data stored in each cell is always the logical “AND” of the new (write data) and the existing data[3].

Interleave access

Data bandwidth of a NAND Flash memory system is dominated by the (1) data transfer time, and (2) page access time. Data transfer time determines the performance of the NAND device when it is actively transferring data. Page access time determines device latency. For read access, the page access time is the time between the read command and the time read data can be shifted out. Read access time ranges from 20us to 50us for most devices. For write access, the access time is the page program time. It is defined as the time between a program command and the time new data of the next page can be shifted in. It is several hundred microseconds for SLC and several milliseconds for MLC. Page program time is much longer than data transfer time (Fig. 6).

Interleaving technique is introduced to reduce the overall access time and to improve performance. Most NAND Flash device that features two planes supports interleave read and interleave write. In interleave read, the NAND Flash allows a page in the second plane to be opened while data of a page in the first plane is being transferred. In interleave write, the NAND Flash device allows two pages from opposite planes to be programmed at the same time. Thus there is only one program time overhead for two pages of data, program time reduced by half.

External interleave can also be employed for the further performance improvement. In this scheme, multiple NAND Flash devices are connected to the same data and control signals of the controller. Each device connection differs only in the chip enable (CE#) signal. When one NAND Flash device is in the program state, the controller can select a second device and transfer data to it. The same overlapping access can be extended to the third and fourth devices sharing the same data pins. The programming time of one chip can overlap with the data transfer and programming time of another chip, allowing the data transfer from the controller to be continuous (Fig. 7).
IV. NAND FLASH DEVICE INTERFACE

Depending on the type of NAND Flash devices, external interface to the NAND Flash device can be synchronous (synchronized to the clock signal) or asynchronous (through the use or read and write pulses).

Let’s take a closer look at the NAND interface evolution from Legacy Asynchronous SDR to Toggle more DDR NAND Flash (Fig. 8)[4].

Asynchronous NAND Flash Interface

Traditional NAND Flash memory uses asynchronous interface to transfer data externally. This means that interface does not use clock signal. Similar to asynchronous SRAM device, asynchronous NAND Flash uses the rising edge of the read and write signals to shift out and shift in data. Even though simple to use, interface is slow and the full bandwidth is not realized. For example, the fastest asynchronous NAND Flash interface has a transfer time of 25ns, with the typical requirements of 12ns read pulse low time and 12ns of read pulse high time. Unfortunately, nearly all controller logic design are synchronous to clock input and pulse width are generated as multiples of clock cycle. If the controller operates at 100Mz (10 ns cycle time), for a pulse width 12ns generation an expensive analog circuit is needed. Digital circuit with this clock rate can only generate pulse width that is a multiple of the cycle time (10ns). So it would generate pulse with 20ns high time and 20ns low time, resulting in 40ns cycle time and 37% reduction from the maximum bandwidth.

At 25ns fastest access time, the maximum bandwidth of asynchronous NAND Flash interface is 40Mbit/sec per pin.

Toggle Mode DDR NAND Flash

Toggle Mode NAND Flash works by generating I/O signals on both the rising and the falling edge of the write enable and read enable signals. With the additional I/O signals, Toggle Mode NAND Flash can achieve the greater data transfer speeds. The bi-directional data query strobe (DQS) pin, which is a change in Toggle Mode NAND from legacy NAND, accesses the I/O pins, allowing for use of both the rising and falling edge of the signal to transfer data[6].

For the SDR NAND, the write enable pin shifts “low”, allowing it to capture the data from I/O pin. A similar process occurs with the read operation, where the read enable pin shifts “low”, allowing it to collect the data from the I/O pin. In the Toggle Mode, however, the write enable pin remains “high”, allowing it to use both edges of the DQS signal for writing data, using two I/O pins. The read process occurs in a similar fashion, where the read enable pin remains “high”, collecting data on both edges of the DQS signal from a pair of I/O pins.

In addition to faster transfer speeds, Toggle Mode NAND requires less power to operate. For legacy NAND Flash technologies, the I/O voltage is 3.3V, but Toggle Mode NAND flash can operate with voltages as low as 1.8V. For a comparison between NAND Memory performances please refer to Figure 9[6]. As we can see from the chart, Toggle mode supports a cycle between 15 and 25 nanoseconds, while legacy NAND is limited to 25ns.

Synchronous NAND Flash

Synchronous NAND Flash utilizes the clock signal to transfer data. Double Data Rate (DDR) interface is also used to transfer 2 bits of data per clock cycle. Synchronous NAND Flash operating at 100Mhz is capable of transferring 200Mbit/sec per pin.

V. NAND FLASH CONTROLLER

What does a NAND Controller Do?

The controller serves as a bridge between the CPU and the NAND Flash Device.

Increased Performance

Approximately three percent of the overall flash array is reserved as a “spare area” in order to cope with flash vulnerabilities, like bit-flipping and bad blocks. Typically from three to six bytes in the spare area are reserved for error detection and correction algorithms, while the remainder of the spare area is used for remapping bad blocks. Without a NAND controller the algorithms that handle these functions can be performed upon the general purpose CPU. However, NAND controllers are available that will perform these functions in...
hardware, thus reduce the CPU load on the device, consequently increasing battery life as well as improving performance.

An advent of Multi-Level Cell (MLC) NAND flash Technology requires even more rigorous error detection algorithms due to the increased error rate.

DMA (Direct Memory Access) function is also found in many controller design. Due to the long page open and program time of the device, using DMA to access NAND Flash is a common technique to let CPU to handle other important system tasks. With DMA, CPU initiates transfer, does other operations while transfer is in progress, and receives interrupt from DMA controller when the operation is done.

**Fast Integration**

The serialized data interface to NAND complicates data transfer in and out of the chip. Unlike NOR or DRAM, an address must be fed in a bit at a time, at just the right time, and then data are read or written in a similar fashion. NAND controller encapsulates the interface with the NAND chip and handles this communication.

Integrated NAND Controllers are becoming popular as mobile processor vendors build NAND Controllers directly into their processors. Following this approach we achieve smallest size and minimal cost design. Disadvantage is an inevitable delay between a new NAND Flash technology appearing and the release of an integrated processor that supports it.

A dedicated, or external, NAND Controller is one in which the NAND controller resides in a separate chip that can be sourced independently of other parts. There is a wide range of latest and greatest NAND Flash chips available. The disadvantage is that chip uses additional PCB space and will cost more money than the integrated approach[2].

A well designed controller delivers the maximum bandwidth of the NAND Flash device while a poorly designed controller not only reduces system performance but also puts a burden on the software design.

**Example of typical NAND Flash Memory IP Core**

**A. Introductive description**

For better understanding, let’s look into the details of a typical NAND Flash Controller IP Core. Chosen configuration was issued by CAST and implemented with ASIC and FPGA approaches. Two main FPGA manufacturers – Xilinx and Altera accepted the challenge.

**Fig. 10 NAND Flash Controller clock diagram**

**Impl****ments a flexible ONFI 2.2 compliant controller for high-capacity MLC, SLC and High-Speed NAND Flash Memory. Works with any suitable memory device with 512 Bytes to 32KB page sizes supporting the Open NAND Flash Interface Working Group (ONFI) standards.**

The Controller offers two error code correction (ECC) mechanisms from the relatively simple single-bit Hamming Code to more sophisticated high-speed BCH ECC. BCH targets applications with high-density memory as well as direct boot from the NAND flash device.

This sixth-generation product are built on silicon-proven previous versions of the controller. Developed for reuse in ASICs and FPGAs, the core is fully synchronous with positive-edge clocking, has no internal three-state buses, and uses a synchronous reset so scan insertion is straightforward. The included verification package features bus models for the AHB master and NAND flash devices to help designers verify the functioning of the core.

**B. Controller Features**

For full list of offered Controller Features please refer to Cast NANDFLASH-CTRL white paper [10].

**C. Functional Description**

For the controller block diagram please refer to Fig. 10.

**DCU**

The Design Control Unit controls all other modules based on the SFR values and current controller state. The main tasks of this module are:

1. Provide enable/disable signal to the DMA and SIU units when they try to get access the FIFO module. Only one of the two units can be active at time.
2. Enable/Disable the ECC module.
3. Provide control signals to the NCU unit.
4. Execute the boot sequence.
5. Interrupt controller.

**SIU**

Opens a window in the address space where the BUFFER and all SFRs are visible, providing access to these elements. Works as glue logic between the system interface and internal controller bus, coordinating their interaction. It is responsible for generating the internal request signal if the controller buffer must be read directly using the controller interface, and it holds transmission on the external bus if access can’t be granted.

**DMA**

Speeds up data transfer between a device on the system bus and the memory, and decreases system bus burden.
FIFO
A 32-bit width asynchronous FIFO module which facilitates transferring data between the input module and NCU when the command sequence is executed.

NCU
The NAND Control Unit is responsible for generation of the NAND flash device access sequences.

ECC
This is an error correction code calculator and a correction unit. A correction word is calculated for each 512B sub page of the NAND Flash memory page.

PHY
This module provides the DDR data interface for the new High Speed devices.

Table 1 shows sample ASIC results for the core optimized with the core constraint of 100 MHz. Number of gates varies from 50K to 120K for the Full Core Configuration.

Table 2 contains a comparison of implementation results for Altera and Xilinx, with Altera achieving a better performance. Results were carried out for a typical configuration, 4 devices per bank, support for high-speed devices, BCH 8, DMA, Soft PHY and memories, and optimized for speed. I/Os are assumed to be routed off-chip.

Table 2 Sample Xilinx and Altera FPGA Results.

<table>
<thead>
<tr>
<th>Altera Device</th>
<th>LEs/ALUTs</th>
<th>Memory</th>
<th>I/Os</th>
<th>Fmax (MHz)</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclone III EP3C153-8</td>
<td>7,593</td>
<td>4 Mbits</td>
<td>256</td>
<td>100/165</td>
<td>9.1</td>
</tr>
<tr>
<td>Cyclone IV EP4C153-8</td>
<td>7,593</td>
<td>4 Mbits</td>
<td>256</td>
<td>110/165</td>
<td>9.1</td>
</tr>
<tr>
<td>Stratix III EP2S15-2</td>
<td>4,095</td>
<td>4 Mbits</td>
<td>256</td>
<td>160/125</td>
<td>9.1</td>
</tr>
<tr>
<td>Stratix IV EP4S15-2</td>
<td>4,095</td>
<td>4 Mbits</td>
<td>256</td>
<td>170/145</td>
<td>9.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Xilinx Device</th>
<th>Slices</th>
<th>Memory</th>
<th>I/Os</th>
<th>Performance (Gbits/clk)</th>
<th>ISE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3E 3S120T-5</td>
<td>5,173</td>
<td>3 BRAM</td>
<td>266</td>
<td>66/145</td>
<td>12.4</td>
</tr>
<tr>
<td>Spartan-3E 6S120T-2</td>
<td>1,812</td>
<td>3 BRAM</td>
<td>266</td>
<td>90/145</td>
<td>12.4</td>
</tr>
<tr>
<td>Virtex-4 5VX2065-3</td>
<td>2,407</td>
<td>3 BRAM</td>
<td>266</td>
<td>125/100</td>
<td>12.4</td>
</tr>
<tr>
<td>Virtex-6 8VX207-2</td>
<td>1,008</td>
<td>2 BRAM</td>
<td>266</td>
<td>190/120</td>
<td>12.4</td>
</tr>
</tbody>
</table>

VI. THE ONFI STANDARD
The lack of standardization among NAND flash manufacturers has been problematic throughout its brief history. Formed in 2006, Open NAND Flash Interface Working Group is a consortium of NAND Flash technology companies who seek to create an industrial standard for NAND Flash device interface. The ONFI standards encompass device pinout, device function, command set, timing, electrical specifications and others. The ONFI consortium includes many major NAND Flash manufacturers with the exception of Samsung and Toshiba.

Historically, different NAND Flash manufacturers produced NAND Flash devices that were very similar in pinout and functionality to each other. Even without a common, industry-wide standard, NAND Flash devices from different manufacturers can often be used interchangeably. As new features are added by each manufacturer, however, the need for a common standard becomes apparent. When the ONFI standard was first published, it formalize the de facto standard then in use. Significant changes to NAND Flash interface such as synchronous and DDR interface was introduced in later revisions of the standard. The Table 3 lists the major changes in different ONFI revision[3].

Challenge of designing ONFI 3.0
Since its inception in 2006, ONFI standard has evolved from asynchronous SDR to a synchronous DDR, running at as fast as 400 MTr/s.

Released on March 15, 2011, ONFI 3.0 introduces NV-DDR2 interface enabling 400MT/s though introduction of number of innovative features, such as differential signalling for DQS and RE, addition of external VrefQ, On-Die-Termination and DQS latency adjustments.

Figure 11 shows some of the performances, calculated Based on MLC and SLC architectures for 400 MTran/s on the interface. These numbers are based on Cache Read and Program with the 16kByte page Dual Plane Max sequential performance with no controller overhead. As can be seen, more LUNs per channel mean the better performance. Reads saturate the channels pretty quickly, not reaching 400 MTran/s mark.

Table 4 is the summary of the features added in ONFI 3.0. Legacy interfaces have been renamed. For further details of ONFI 3.0 please refer to [5].

Table 3 Major changes in different ONFI revisions

<table>
<thead>
<tr>
<th>ONFI Revision</th>
<th>Important Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONFI 1.0</td>
<td>Asynchronous interface only, maximum 300Mbps data rate.</td>
</tr>
<tr>
<td>ONFI 2.2</td>
<td>Add synchronous interface, maximum data rate is 200Mbps.</td>
</tr>
<tr>
<td>ONFI 2.3</td>
<td>Built-in ECC</td>
</tr>
<tr>
<td>ONFI 2.4</td>
<td>Maximum synchronous transfer rate is 400Mbps, asynchronous DDR transfer rate is 200Mbps, differential signal, on-die termination</td>
</tr>
</tbody>
</table>
VII. OVERVIEW OF SPECIFIC NAND FLASH CONTROLLER FEATURES

A. Error Correction

Next generation flash parts will continue to evolve to smaller technologies (32, 19 nm and beyond), more bits per cell and larger page size. As a bit per cell gets smaller, fewer electrons can be trapped in the floating gates, leading to more uncertainty in the amount of charge present. The effect of narrowing the valid voltage ranges for a given value, increases the chances for program and read disturbance to corrupt data.

Error correction is an integral part of using NAND flash that ensures data integrity. The ECC requirement ranges from 1-bit correction per 512 bytes of data to 40-bit correction per 1 kbyte of data.

When data is written to each page, the ECC for the page is computed and stored into the spare column of the page. When data is read from the page, both the data page and spare column are read. ECC is then computed and error rate corrected. As a result, the entire page of data has to be read or written in order to compute ECC.

Any discussion on the strength of error correction algorithms comes down to the raw bit error rate of the underlying medium and the allowable block error rate that is acceptable to an application. The application’s block error rate can be computed from the bit error rate fairly easily:

\[ \text{BLER} = \frac{N}{E} \times p^E \times (1-p)^{N-E} \] (1)

where \( N \) is the number of bits per block, \( E \) is the number of errors in a block and \( p \) is the bit error rate.

Equation 1 basically states that the block error rate is dependent on the number of statistical combinations of failing bit patterns, the probability of \( E \) errors occurring, and the probability of \( N-E \) correct data bits.

Most frequently used by designers are Reed-Solomon (R/S) and Bose-Chaudhuri-Hocquenghem (BCH) algorithms. While both of them are similar, R/S codes perform correction over multi-bit symbols while BCH performs correction over single-bit symbols. Typically R/S codes are used when errors are expected to occur in bursts and BCH is used when the bit errors are randomly distributed.

Using the BLER equation, it is possible to graph the relative strengths of the various ECC schemes to determine statistically which ones are optimal. Figure 12 presents a graph, bit error rate shown on horizontal axis, corresponding block error rate on the vertical axis for various ECC schemes for the high bit error rates.

Note that it becomes obvious that ECC20 over 1KB blocks is roughly equivalent to ECC16 over 512 bytes, but is far stronger as the bit error rate drops. At a BER of 1E-4, moving from ECC20 to ECC24 provides roughly 6 orders of magnitude greater protection over either of these schemes[7].

B. Bad Block Management

Bad blocks are those blocks that contain one or more invalid bits whose reliability is not guaranteed. Bad blocks may be present when the device is shipped, or may develop during its lifetime. Note that a bad block does not affect the performance of valid blocks because it is isolated from the bit line and common source line by a select transistor.

NAND Flash devices are supplied with all the locations inside valid blocks erased. The bad block information is written prior to shipping. To allow the system to recognize the bad blocks based on the original information, can use algorithm from the Figure 13. Once created, the bad block table is saved to a good block that on rebooting the NAND Flash the table is loaded into RAM. Blocks contained in the bad block table are not addressable, if FTL addresses one of the bad blocks, software redirects it to a good block.

As the failure of the page programming does not affect data in other pages, the block can be replaced by reprogramming the current data and copying the rest of replaced block to an available valid block. Blocks are marked as bad and new blocks...
allocated using two general methods: Skip block and Reverse block.

**Skip Block Method**

In the skip block method, the algorithm creates the bad block table and when the target address corresponds to a bad address, the data is stored in the next good block, skipping the bad block.

When the bad block is generated during the lifetime of the NAND Flash device, its data is also stored in the next good block.

**Reverse Block Method**

According to this method, bad blocks are not skipped but replaced by good blocks by redirecting FTL to a known free good block. For that purpose, reserved block area is created, in addition to user addressable area. FTL uses the user addressable block area to store data whereas the reserved block area is only used for bad block replacement and remapping of the developed bad blocks.

Each time the FTL writes a logical sector, it calculates the physical address of the block to which it will write. Then, before the FTL starts writing, the bad block management software checks whether the block is bad or not. If its bad, it returns the address of the good block to which sector is remapped. Note that this mechanism is thus transparent to FTL[8].

**C. Wear Levelling**

NAND Flash cells have a limited lifetime. The specification for SLC is set to 100K W/E cycles, and for MLC from 5K to 10K E/W cycle. If a particular page of data is updated often, cells within it become useless quickly, rendering the entire system to fail. Wear levelling is a technique that remaps the same logic address to different physical page each time it is used. Its goal is to use the entire NAND Flash device evenly, maximizing its lifespan. Wear levelling and consequent bad block management are typically done by the Flash Transaction Layer (FTL) software (for the software tool chain please refer to Fig. 14).

In most cases of the wear-levelling implementation, controller maintains the lookup table to translate the memory array physical block address (PBA) to the logical block address (LBA) used by the host system (Fig. 15).

Depending on the wear-levelling method used, controller typically either writes to the available erased block with the lowest overall erase count (dynamic wear leveling) or selects an available target block with the lowest overall count, erases the block if necessary, writes new data in it (static wear leveling).

**Implementing Dynamic Wear Leveling**

Dynamic wear levelling is a method of pooling the available blocks that are free of data and selecting the block with the lowest erase count for the next write. The method is optimal for dynamic data because only the non-static part of the array is wear-leveled.

As an example, consider the device with a 25%/75% split of dynamic data versus static data, respectively. Thus dynamic wear levelling targets the 25% of the array. In this case, 25%
of the available blocks are used to their maximum cycle (Fig. 16).

Implementing Static Wear Leveling

This technique utilizes all good blocks to evenly distribute wear. It is done by tracking the cycle count of all good blocks and attempting to evenly distribute block wear though the entire device. Blocks that contain static data with erase counts that begin to lag are included in the wear-leveling block pool, with the data moved to blocks with higher erase counts.

An additional step of moving static data to free up space usually slows down the write performance, however, static wear levelling is proved to be better than dynamic (Fig. 16). Advantages and disadvantages of wear levelling techniques are listed in Table 5 [9].

VIII. CONCLUSIONS

We went through the survey about high speed interfacing NAND Flash Memory, NAND Flash array architecture and structure of NAND Flash Controller with emphasis on its specific features such as EEC, Bad Block Management and Wear Leveling.

IX. REFERENCES


Table 5 Static vs Dynamic Wear-Leveling

<table>
<thead>
<tr>
<th>Wear-Leveling Method</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>
| Static               | Maximizes device life  
                      | Most robust wear-leveling method  
                      | Most efficient use of memory array | Requires more controller overhead  
                      | Can slow WRITE operations  
                      | Higher power consumption  
                      | More complicated to implement than dynamic wear leveling |
| Dynamic              | Improves device life over no wear leveling at all  
                      | Easier to implement than static wear leveling  
                      | No impact on device performance | May not optimize device life |