Introduction to Flash Memory (T1A)

Jim Cooke (jcooke@micron.com)
Staff Architect and Technologist, Architecture Development Group
Micron Technology, Inc.
Agenda

• The basics of Flash and NAND
  – Flash cell comparison
  – NAND and NOR attributes and interface comparison
  – Detailed operations
    – Commands, address, and data operations

• Connecting NAND to a RISC or DSP processor

• More NAND Flash device detail
  – SLC vs. MLC
  – All NAND devices are not created equal
    – Architecture, features, and performance comparisons

• Performance bottlenecks

• ONFI and high-speed NAND introduction

• NAND error modes
  – Program disturb
  – Read disturb
  – Data retention
  – Endurance
  – Wear-leveling
  – ECC fixes almost everything
A Quick Review of Flash Basics

- Cell differences
- NAND attributes
- NAND vs. NOR
Flash Basics

- Flash data is grouped into blocks, which are the smallest erasable entity
  - Erasing a block sets all bits to “1” or bytes to FFh
- The programming operation changes erased bits from “1” to “0”
  - The smallest entity that can be programmed is a bit
- While NAND cannot inherently perform random access, it is possible at the system level through shadowing
Flash Memory Cell Comparison

<table>
<thead>
<tr>
<th></th>
<th>NAND</th>
<th>NOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell array</td>
<td><img src="image" alt="NAND Cell" /></td>
<td><img src="image" alt="NOR Cell" /></td>
</tr>
<tr>
<td></td>
<td>word line</td>
<td>word line</td>
</tr>
<tr>
<td></td>
<td>source line</td>
<td>source line</td>
</tr>
<tr>
<td></td>
<td>Unit Cell</td>
<td>Unit Cell</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="NAND Layout" /></td>
<td><img src="image" alt="NOR Layout" /></td>
</tr>
<tr>
<td>Layout</td>
<td>2F</td>
<td>5F</td>
</tr>
<tr>
<td>Cross-section</td>
<td><img src="image" alt="NAND Cross" /></td>
<td><img src="image" alt="NOR Cross" /></td>
</tr>
<tr>
<td>Cell size</td>
<td>4F²</td>
<td>10F²</td>
</tr>
</tbody>
</table>

- NAND Flash’s small cell size enables high density and low cost
Basic NAND Attributes

- NAND is very similar to a disk drive; it is sector-based (page-based) Flash and is well-suited for storage of sequential data (such as pictures, audio, and files)
  - Like a disk drive, NAND is not well-suited for random access, such as executing code, although random access can be accomplished at the system level by shadowing the data to RAM (similar to what a PC does with BIOS)
  - Like a disk drive, NAND devices have bad sectors or blocks and require management
  - Like a disk drive, NAND requires error correction code (ECC)
  - Unlike a disk drive, it is possible to wear out the NAND cell; with good wear-leveling, this is typically not an issue
Basic NAND Attributes

- NAND is available in large capacities and is the lowest cost Flash memory available today

- NAND is finding its way into many embedded applications and is used in virtually all removable cards
  - USB cards
  - Memory stick
  - MMC multimedia card
  - SD secure digital
  - CF compact Flash

- Multiplexed interface provides similar pinout over all devices
  - x8 signal pinout has not changed from 64Mb pinout

- x8 devices are used mostly in high capacity (3.3V) consumer applications; the x16 devices are mostly used in embedded (1.8V) applications
## Basic NAND/NOR Comparison

<table>
<thead>
<tr>
<th>NAND</th>
<th>NOR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advantages</strong></td>
<td><strong>Advantages</strong></td>
</tr>
<tr>
<td>• Fast writes</td>
<td>• Random access</td>
</tr>
<tr>
<td>• Fast erases</td>
<td>• Byte writes possible</td>
</tr>
<tr>
<td><strong>Disadvantages</strong></td>
<td><strong>Disadvantages</strong></td>
</tr>
<tr>
<td>• Slow random access</td>
<td>• Slow writes</td>
</tr>
<tr>
<td>• Byte writes difficult</td>
<td>• Slow erase</td>
</tr>
<tr>
<td><strong>Applications</strong></td>
<td><strong>Applications</strong></td>
</tr>
<tr>
<td>• File (disk) applications</td>
<td>• Replacement of EPROM</td>
</tr>
<tr>
<td>• Voice, data, video recorder</td>
<td>• Execute directly from nonvolatile memory</td>
</tr>
<tr>
<td>• Any large sequential data</td>
<td></td>
</tr>
</tbody>
</table>
## Flash Memory Comparison

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>NAND Flash MT29F2G08</th>
<th>NOR Flash MT28F128J3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random access read</td>
<td>25µs (first byte)</td>
<td>0.12µs</td>
</tr>
<tr>
<td></td>
<td>0.03µs each for remaining 2,111 bytes</td>
<td></td>
</tr>
<tr>
<td>Sustained read speed (sector basis)</td>
<td>23 MB/s (x8) or 37 MB/s (x16)</td>
<td>20.5 MB/s (x8) or 41 MB/s (x16)</td>
</tr>
<tr>
<td>Random write speed</td>
<td>~300µs/2,112 bytes</td>
<td>180µs/32 bytes</td>
</tr>
<tr>
<td>Sustained write speed (sector basis)</td>
<td>5 MB/s</td>
<td>0.178 MB/s</td>
</tr>
<tr>
<td>Erase block size</td>
<td>128KB</td>
<td>128KB</td>
</tr>
<tr>
<td>Erase time per block (typ)</td>
<td>2ms</td>
<td>750ms</td>
</tr>
</tbody>
</table>

NAND Flash is ideal for file storage, such as data or image files; if code is stored, it must be shadowed to RAM first, as in a PC.

NOR Flash is ideal for direct code execution (boot code) although it still needs to be shadowed (for speed).
Flash Interface Comparison

- **NOR Flash**
  - Random-access interface typically composed of:
    - CE# — chip enable
    - WE# — write enable
    - OE# — output enable
    - D15-D0 — data bus
    - A20-A0 — address bus
    - WP# — write protect

- **NAND Flash**
  - I/O device-type interface composed of:
    - CE# — chip enable
    - WE# — write enable
    - RE# — read enable
    - CLE — command latch enable
    - ALE — address latch enable
    - I/O 7-0 — data bus (I/O 15-0 for x16 parts)
    - WP# — write protect
    - R/B# — ready/busy

41 pins

23 pins (for x16)
NAND Flash Physical Interface (TSOP 1)

Indirect addressing enables no pinout changes among densities

Note 2: Additional Vcc and Vss recommended for new PCB designs
NAND Block Diagram

<table>
<thead>
<tr>
<th>ALE</th>
<th>CLE</th>
<th>NAND Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Data register</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Command register</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Address register</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Undefined</td>
</tr>
</tbody>
</table>
Detailed Operations

Architecture
Addressing
Basic commands
Basic NAND Flash Operations

Program:
- Data input
- Page Block
- Page-based operation

Register

Read:
- High speed
- Serial read
- Page-based operation

Erase:
- Block erase

Multiplexed command, address, data protocol

NAND controller

R/B is open drain and requires a pull-up resistor

Multiplexed command, address, data protocol

Command input

Address input (5 cycles)
SLC NAND Flash Memory Diagram

- Serial input: (x8 or x16) ~30ns (clk)
- Program: ~300ms/page
- Serial output: (x8 or x16) ~30ns (clk)
- Read (page load): 25ms
- Block erase ~2ms

Data Area (2,048 bytes)

NAND Memory Array

NAND Page 2,112 bytes

NAND Block

64 pages per block

2,048 blocks (2Gb device)

8-bit byte or 16-bit word

Spare Area (ECC, etc.) (64 bytes)
Basic Access
Large-Block NAND Addressing

```
<table>
<thead>
<tr>
<th>Cycle</th>
<th>I/O7</th>
<th>I/O6</th>
<th>I/O5</th>
<th>I/O4</th>
<th>I/O3</th>
<th>I/O2</th>
<th>I/O1</th>
<th>I/O0</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>CA7</td>
<td>CA6</td>
<td>CA5</td>
<td>CA4</td>
<td>CA3</td>
<td>CA2</td>
<td>CA1</td>
<td>CA0</td>
</tr>
<tr>
<td>Second</td>
<td>LOW</td>
<td>LOW</td>
<td>LOW</td>
<td>LOW</td>
<td>CA11</td>
<td>CA10</td>
<td>CA9</td>
<td>CA8</td>
</tr>
<tr>
<td>Third</td>
<td>RA19</td>
<td>RA18</td>
<td>RA17</td>
<td>RA16</td>
<td>RA15</td>
<td>RA14</td>
<td>RA13</td>
<td>RA12</td>
</tr>
<tr>
<td>Fourth</td>
<td>RA27</td>
<td>RA26</td>
<td>RA25</td>
<td>RA24</td>
<td>RA23</td>
<td>RA22</td>
<td>RA21</td>
<td>RA20</td>
</tr>
<tr>
<td>Fifth</td>
<td>LOW</td>
<td>LOW</td>
<td>LOW</td>
<td>LOW</td>
<td>LOW</td>
<td>LOW</td>
<td>RA29</td>
<td>RA28</td>
</tr>
</tbody>
</table>
```

Notes: 1. Die address boundary: 0 = 0 – 2Gb, 1 = 2Gb – 4Gb.
## NAND Command List

**Standard 2Gb (256MB) NAND**

<table>
<thead>
<tr>
<th>Command</th>
<th>Command Cycle 1</th>
<th>Number of Address Cycles</th>
<th>Data Cycles Required&lt;sup&gt;1&lt;/sup&gt;</th>
<th>Command Cycle 2</th>
<th>Valid During Busy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE READ</td>
<td>00h</td>
<td>5</td>
<td>No</td>
<td>30h</td>
<td>No</td>
</tr>
<tr>
<td>PAGE READ CACHE MODE START</td>
<td>31h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>PAGE READ CACHE MODE START LAST</td>
<td>3Fh</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>READ for INTERNAL DATA MOVE</td>
<td>00h</td>
<td>5</td>
<td>No</td>
<td>35h</td>
<td>No</td>
</tr>
<tr>
<td>RANDOM DATA READ</td>
<td>05h</td>
<td>2</td>
<td>No</td>
<td>E0h</td>
<td>No</td>
</tr>
<tr>
<td>READ ID</td>
<td>90h</td>
<td>1</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>READ STATUS</td>
<td>70h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>Yes</td>
</tr>
<tr>
<td>PROGRAM PAGE</td>
<td>80h</td>
<td>5</td>
<td>Yes</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>PROGRAM PAGE CACHE MODE</td>
<td>80h</td>
<td>5</td>
<td>Yes</td>
<td>15h</td>
<td>No</td>
</tr>
<tr>
<td>PROGRAM for INTERNAL DATA MOVE</td>
<td>85h</td>
<td>5</td>
<td>Optional</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>RANDOM DATA INPUT</td>
<td>85h</td>
<td>2</td>
<td>Yes</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>BLOCK ERASE</td>
<td>60h</td>
<td>3</td>
<td>No</td>
<td>D0h</td>
<td>No</td>
</tr>
<tr>
<td>RESET</td>
<td>FFh</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>Yes</td>
</tr>
</tbody>
</table>

For ease of presentation:  
**Basic command** | **Advanced command**
## Reset Operation

<table>
<thead>
<tr>
<th>Command</th>
<th>Command Cycle 1</th>
<th>Number of Address Cycles</th>
<th>Data Cycles Required</th>
<th>Command Cycle 2</th>
<th>Valid During Busy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE READ</td>
<td>00h</td>
<td>5</td>
<td>No</td>
<td>30h</td>
<td>No</td>
</tr>
<tr>
<td>PAGE READ CACHE MODE START</td>
<td>31h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>PAGE READ CACHE MODE START LAST</td>
<td>3Fh</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>READ for INTERNAL DATA MOVE</td>
<td>00h</td>
<td>5</td>
<td>No</td>
<td>35h</td>
<td>No</td>
</tr>
<tr>
<td>RANDOM DATA READ</td>
<td>05h</td>
<td>2</td>
<td>No</td>
<td>E0h</td>
<td>No</td>
</tr>
<tr>
<td>READ ID</td>
<td>90h</td>
<td>1</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>READ STATUS</td>
<td>70h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>Yes</td>
</tr>
<tr>
<td>PROGRAM PAGE</td>
<td>80h</td>
<td>5</td>
<td>Yes</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>PROGRAM PAGE CACHE MODE</td>
<td>80h</td>
<td>5</td>
<td>Yes</td>
<td>15h</td>
<td>No</td>
</tr>
<tr>
<td>PROGRAM for INTERNAL DATA MOVE</td>
<td>85h</td>
<td>5</td>
<td>Optional</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>RANDOM DATA INPUT</td>
<td>85h</td>
<td>2</td>
<td>Yes</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>BLOCK ERASE</td>
<td>60h</td>
<td>3</td>
<td>No</td>
<td>D0h</td>
<td>No</td>
</tr>
<tr>
<td>RESET</td>
<td>FFh</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Diagram:**

1. CLE
2. CE#
3. WE#
4. R/B#
5. Vdd
6. VOx
7. Reset Command

---

Santa Clara, CA  USA
August 22–24, 2008
Read ID Operation

<table>
<thead>
<tr>
<th>Command</th>
<th>Command Cycle 1</th>
<th>Number of Address Cycles</th>
<th>Data Cycles Required</th>
<th>Command Cycle 2</th>
<th>Valid During Busy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE READ</td>
<td>00h</td>
<td>5</td>
<td>No</td>
<td>30h</td>
<td>No</td>
</tr>
<tr>
<td>PAGE READ CACHE MODE START</td>
<td>31h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>PAGE READ CACHE MODE START LAST</td>
<td>3Fh</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>READ for INTERNAL DATA MOVE</td>
<td>00h</td>
<td>5</td>
<td>No</td>
<td>35h</td>
<td>No</td>
</tr>
<tr>
<td>RANDOM DATA READ</td>
<td>05h</td>
<td>2</td>
<td>No</td>
<td>E0h</td>
<td>No</td>
</tr>
<tr>
<td>READ ID</td>
<td>90h</td>
<td>1</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>READ STATUS</td>
<td>70h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>Yes</td>
</tr>
<tr>
<td>PROGRAM PAGE</td>
<td>80h</td>
<td>5</td>
<td>Yes</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>PROGRAM PAGE CACHE MODE</td>
<td>80h</td>
<td>5</td>
<td>Yes</td>
<td>15h</td>
<td>No</td>
</tr>
<tr>
<td>PROGRAM for INTERNAL DATA MOVE</td>
<td>85h</td>
<td>5</td>
<td>Optional</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>RANDOM DATA INPUT</td>
<td>85h</td>
<td>2</td>
<td>Yes</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>BLOCK ERASE</td>
<td>60h</td>
<td>3</td>
<td>No</td>
<td>D0h</td>
<td>No</td>
</tr>
<tr>
<td>RESET</td>
<td>FFh</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>Yes</td>
</tr>
</tbody>
</table>

CLE

CE#

WE#

ALE

RE#

I/Ox

90h READ ID Command

00h Address, 1 Cycle

Byte 0 Man ID¹

Byte 1 Device ID¹

Byte 2 Don’t Care

Byte 3²
## Device IDs

<table>
<thead>
<tr>
<th>Density</th>
<th>x8/x16</th>
<th>1.8V/3.3V</th>
<th># of Die</th>
<th>Byte 0 Manf. ID</th>
<th>Byte 1 Device ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1Gb x8</td>
<td>1.8V</td>
<td>1</td>
<td></td>
<td>2Ch</td>
<td>A1h</td>
</tr>
<tr>
<td>1Gb x8</td>
<td>3.3V</td>
<td>1</td>
<td></td>
<td>2Ch</td>
<td>F1h</td>
</tr>
<tr>
<td>1Gb x16</td>
<td>1.8V</td>
<td>1</td>
<td></td>
<td>2Ch</td>
<td>B1h</td>
</tr>
<tr>
<td>1Gb x16</td>
<td>3.3V</td>
<td>1</td>
<td></td>
<td>2Ch</td>
<td>C1h</td>
</tr>
<tr>
<td>2Gb x8</td>
<td>1.8V</td>
<td>1</td>
<td></td>
<td>2Ch</td>
<td>AAh</td>
</tr>
<tr>
<td>2Gb x8</td>
<td>3.3V</td>
<td>1</td>
<td></td>
<td>2Ch</td>
<td>DAh</td>
</tr>
<tr>
<td>2Gb x16</td>
<td>1.8V</td>
<td>1</td>
<td></td>
<td>2Ch</td>
<td>BAh</td>
</tr>
<tr>
<td>2Gb x16</td>
<td>3.3V</td>
<td>1</td>
<td></td>
<td>2Ch</td>
<td>CAh</td>
</tr>
<tr>
<td>4Gb x8</td>
<td>1.8V</td>
<td>2</td>
<td></td>
<td>2Ch</td>
<td>ACh</td>
</tr>
<tr>
<td>4Gb x8</td>
<td>3.3V</td>
<td>2</td>
<td></td>
<td>2Ch</td>
<td>DCh</td>
</tr>
<tr>
<td>4Gb x16</td>
<td>1.8V</td>
<td>2</td>
<td></td>
<td>2Ch</td>
<td>BCh</td>
</tr>
<tr>
<td>4Gb x16</td>
<td>3.3V</td>
<td>2</td>
<td></td>
<td>2Ch</td>
<td>CCh</td>
</tr>
<tr>
<td>8Gb x8</td>
<td>1.8V</td>
<td>4</td>
<td></td>
<td>2Ch</td>
<td>ACh</td>
</tr>
<tr>
<td>8Gb x8</td>
<td>3.3V</td>
<td>4</td>
<td></td>
<td>2Ch</td>
<td>DCh</td>
</tr>
<tr>
<td>8Gb x16</td>
<td>1.8V</td>
<td>4</td>
<td></td>
<td>2Ch</td>
<td>BCh</td>
</tr>
<tr>
<td>8Gb x16</td>
<td>3.3V</td>
<td>4</td>
<td></td>
<td>2Ch</td>
<td>CCh</td>
</tr>
</tbody>
</table>
## Read Status Operation

<table>
<thead>
<tr>
<th>Command</th>
<th>Command Cycle 1</th>
<th>Number of Address Cycles</th>
<th>Data Cycles Required</th>
<th>Command Cycle 2</th>
<th>Valid During Busy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE READ</td>
<td>00h</td>
<td>5</td>
<td>No</td>
<td>30h</td>
<td>No</td>
</tr>
<tr>
<td>PAGE READ CACHE MODE START</td>
<td>31h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>PAGE READ CACHE MODE START LAST</td>
<td>3Fh</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>READ for INTERNAL DATA MOVE</td>
<td>00h</td>
<td>5</td>
<td>No</td>
<td>35h</td>
<td>No</td>
</tr>
<tr>
<td>RANDOM DATA READ</td>
<td>05h</td>
<td>2</td>
<td>No</td>
<td>E0h</td>
<td>No</td>
</tr>
<tr>
<td>READ ID</td>
<td>90h</td>
<td>1</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>READ STATUS</td>
<td>70h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>Yes</td>
</tr>
<tr>
<td>PROGRAM PAGE</td>
<td>80h</td>
<td>5</td>
<td>Yes</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>PROGRAM PAGE CACHE MODE</td>
<td>80h</td>
<td>5</td>
<td>Yes</td>
<td>15h</td>
<td>No</td>
</tr>
<tr>
<td>PROGRAM for INTERNAL DATA MOVE</td>
<td>85h</td>
<td>5</td>
<td>Optional</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>RANDOM DATA INPUT</td>
<td>85h</td>
<td>2</td>
<td>Yes</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>BLOCK ERASE</td>
<td>60h</td>
<td>3</td>
<td>No</td>
<td>D0h</td>
<td>No</td>
</tr>
<tr>
<td>RESET</td>
<td>FFh</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>Yes</td>
</tr>
</tbody>
</table>

![Timing Diagram](image)
### NAND Flash Read Status Results

<table>
<thead>
<tr>
<th>SR Bit</th>
<th>Program Page</th>
<th>Program Page Cache Mode</th>
<th>Page Read</th>
<th>Page Read Cache Mode</th>
<th>Block Erase</th>
<th>Definition</th>
</tr>
</thead>
</table>
| 0⁴     | Pass/fail    | Pass/fail (N)          | –         | –                    | Pass/fail   | 0 = Successful PROGRAM/ERASE  
1 = Error in PROGRAM/ERASE          |
| 1      | –            | Pass/fail (N-1)        | –         | –                    | –           | 0 = Successful PROGRAM  
1 = Error in PROGRAM               |
| 2      | –            | –                      | –         | –                    | –           | 0                                |
| 3      | –            | –                      | –         | –                    | –           | 0                                |
| 4      | –            | –                      | –         | –                    | –           | 0                                |
| 5      | Ready/busy   | Ready/busy²            | Ready/busy| Ready/busy²         | Ready/busy | 0 = Busy  
1 = Ready                            |
| 6      | Ready/busy   | Ready/busy cache³      | Ready/busy| Ready/busy cache³  | Ready/busy | 0 = Busy  
1 = Ready                            |
| 7      | Write protect| Write protect          | Write protect| Write protect     | Write protect| 0 = Protected  
1 = Not protected                   |

Read status typically = E0h when the NAND is ready with no error
Block Erase Operation

2,112 bytes

NAND Memory Array

NAND Page 2,112 bytes

64 pages per block

NAND Block

Data Area (2,048 bytes)

Spare Area (ECC, etc.) (64 bytes)

8-bit byte or 16-bit word

Block Erase ~2ms

2,048 blocks (2Gb device)
# Block Erase Operation

<table>
<thead>
<tr>
<th>Command</th>
<th>Command Cycle 1</th>
<th>Number of Address Cycles</th>
<th>Data Cycles Required</th>
<th>Command Cycle 2</th>
<th>Valid During Busy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE READ</td>
<td>00h</td>
<td>5</td>
<td>No</td>
<td>30h</td>
<td>No</td>
</tr>
<tr>
<td>PAGE READ CACHE MODE START</td>
<td>31h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>PAGE READ CACHE MODE START LAST</td>
<td>39h</td>
<td>—</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>READ for INTERNAL DATA MOVE</td>
<td>00h</td>
<td>5</td>
<td>No</td>
<td>35h</td>
<td>No</td>
</tr>
<tr>
<td>RANDOM DATA READ</td>
<td>05h</td>
<td>2</td>
<td>Yes</td>
<td>00h</td>
<td>No</td>
</tr>
<tr>
<td>READ ID</td>
<td>96h</td>
<td>1</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>READ STATUS</td>
<td>70h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>Yes</td>
</tr>
<tr>
<td>PROGRAM PAGE</td>
<td>80h</td>
<td>5</td>
<td>Yes</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>PROGRAM PAGE CACHE MODE</td>
<td>80h</td>
<td>5</td>
<td>Yes</td>
<td>15h</td>
<td>No</td>
</tr>
<tr>
<td>PROGRAM for INTERNAL DATA MOVE</td>
<td>85h</td>
<td>5</td>
<td>Optional</td>
<td>00h</td>
<td>No</td>
</tr>
<tr>
<td>RANDOM DATA INPUT</td>
<td>85h</td>
<td>2</td>
<td>Yes</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>BLOCK ERASE</td>
<td>60h</td>
<td>3</td>
<td>No</td>
<td>D0h</td>
<td>No</td>
</tr>
<tr>
<td>RESET</td>
<td>FFh</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Diagram:**

- CLE
- CE#
- WE#
- ALE
- R/B#
- RE#
- I/Ox
- Address Input (3 Cycles)
- D0h
- 70h
- Status

- I/O 0 = 0  ERASE successful
- I/O 0 = 1  ERASE error
Program Operation

- The programming operation can program only "0" bits
- If you don’t want to program a bit, set it to “1”
- The register is automatically loaded with all “1s” by the 80h command (note the 85h command does not do this)
- After a bit has been programmed to a “0,” if you want to turn it back to a “1,” you must complete a block erase to return the entire block back to all “1s”
- Programming must be done sequentially (within a block)

Serial input: (x8 or x16) 30ns (clk)

- NAND Memory Array
- 2,048 blocks (2Gb device)
- Data Area (2,048 bytes)
- Spare Area (ECC, etc.) (64 bytes)
- 8-bit byte or 16-bit word

Register

- NAND Page 2,112 bytes
- 2,112 bytes
- 64 pages per block

Program (‘PROG):
~300µs/page
Program Operation

<table>
<thead>
<tr>
<th>Command</th>
<th>Command Cycle 1</th>
<th>Number of Address Cycles</th>
<th>Data Cycles Required</th>
<th>Command Cycle 2</th>
<th>Valid During Busy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE READ</td>
<td>00h</td>
<td>5</td>
<td>No</td>
<td>38h</td>
<td>No</td>
</tr>
<tr>
<td>PAGE READ CACHE MODE START</td>
<td>21h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>PAGE READ CACHE MODE START LAST</td>
<td>38h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>READ for INTERNAL DATA MOVE</td>
<td>00h</td>
<td>5</td>
<td>No</td>
<td>38h</td>
<td>No</td>
</tr>
<tr>
<td>RANDOM DATA READ</td>
<td>05h</td>
<td>2</td>
<td>No</td>
<td>50h</td>
<td>No</td>
</tr>
<tr>
<td>READ ID</td>
<td>90h</td>
<td>1</td>
<td>No</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>READ STATUS</td>
<td>70h</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>Yes</td>
</tr>
<tr>
<td>PROGRAM PAGE</td>
<td>80h</td>
<td>5</td>
<td>Yes</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>PROGRAM PAGE CACHE MODE</td>
<td>80h</td>
<td>5</td>
<td>Yes</td>
<td>15h</td>
<td>No</td>
</tr>
<tr>
<td>PROGRAM for INTERNAL DATA MOVE</td>
<td>85h</td>
<td>5</td>
<td>Optional</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>RANDOM DATA INPUT</td>
<td>85h</td>
<td>2</td>
<td>Yes</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>BLOCK ERASE</td>
<td>60h</td>
<td>3</td>
<td>No</td>
<td>10th</td>
<td>No</td>
</tr>
<tr>
<td>RESET</td>
<td>FFh</td>
<td>—</td>
<td>No</td>
<td>—</td>
<td>Yes</td>
</tr>
</tbody>
</table>
You can input as many address and data combinations as you want.

The page is programmed when you issue the 10h confirmation. Each of these counts toward the partial-page programming limit (of 8).
Read Operation

- Read transfers the addressed page from the array to the register.
- The column address specifies the first byte out; it can be offset by any amount.
- Each clock (RE#) shifts a byte (or word) out.

Read access (tR): ~25µs/page

Serial output: (x8 or x16) ~30ns clock

Register

2,112 bytes

NAND Page 2,112 bytes

NAND Memory Array

NAND Block

Data Area (2,048 bytes)

Spare Area (ECC, etc.) (64 bytes)

8-bit byte or 16-bit word

64 pages per block
The RANDOM READ command allows you to specify a new two-byte column address.

Can use the RANDOM READ command to jump around anywhere on the page.

You can access random data by inputting a random data read 05h cmd, address, E0h and clock the desired data out.
Partial-Page Programming

- NOP specifies the number of programming operations that can be executed on the same page
- Pages are programmed in groups due to the large page sizes (SLC only)
  - Typical PC sector size is 512, so four PC sectors fit into one 2K page
  - Programming ECC info separately from the data could require an additional four operations
  - The user can have other info (logical mapping or wear-leveling) in the spare area
- It is best to minimize partial-page programming
  - The number of partial-page program operations is the number of complete programming operations (with confirm 10h) to the same location without an erase
- MLC devices have an NOP of 1
Methods for Data and Spare Information Placement

Data and spare information adjacent

- 2,112 bytes
- Data Area (512 bytes)
- Data Area (512 bytes)
- Data Area (512 bytes)
- Data Area (512 bytes)
- Spare Areas (ECC, etc.)
  - 16 bytes each

Data and spare information separate

- 2,048 bytes
- Data Area (512 bytes)
- Data Area (512 bytes)
- Data Area (512 bytes)
- Data Area (512 bytes)
- 64 bytes
Connecting NAND to a RISC Processor or DSP That Does not Include a NAND Controller
Direct Connection to RISC Processor

Memory Mapped NAND Interface

- If microprocessor address 4 is connected to CLE and address 5 is connected to ALE, the NAND can be accessed by a software that uses only three address locations
  - Command register can be accessed by writing to address XX010h
  - Address register can be accessed by writing to address XX020h
  - Data register can be accessed by writing/reading to address XX000h

<table>
<thead>
<tr>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A5</td>
<td>A4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ALE</th>
<th>CLE</th>
<th>Memory Address Offset</th>
<th>NAND Register Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Data register</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Command register</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>Address register</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>Undefined (don’t use)</td>
</tr>
</tbody>
</table>
Glueless Microprocessor NAND Interface

**Program Function**

- **CE**
  - low

- **ALE**
  -

- **CLE**
  -

- **WE**
  -

- **RE**
  - Command
  - D2112

- **I/O1~8**
  - Address Input (5 cycles)
  - Wait (tR) ~300us

Assume CS1 address space is 0xFFF000-0xFFF0FF
Glueless Microprocessor NAND Interface

Pseudo-Code Example for PROGRAM:
(All numbers in HEX)
80 -> FFF010 ; CMD = 80
ColL -> FFF020 ; low column
ColH -> FFF020 ; high column
RowL -> FFF020 ; low ROW
RowM -> FFF020 ; Mid ROW
RowH -> FFF020 ; High ROW
D0 -> FFF000 ; Data 0
D1 -> FFF000 ; Data 1

(Complete remaining data)
D2111 -> FFF000 ; Data 2111
10 -> FFF010 ; CMD = 10

Program Function

<table>
<thead>
<tr>
<th>CE</th>
<th>low</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALE</td>
<td></td>
</tr>
<tr>
<td>CLE</td>
<td></td>
</tr>
<tr>
<td>WE</td>
<td></td>
</tr>
<tr>
<td>RE</td>
<td></td>
</tr>
</tbody>
</table>

Address Input (5 cycles)

Wait (tR) ~220us

Next operation

D2111

I/O1~8

D0 D1 D2 D3 D4 D5... 10h Command

Command

80h

Col Col Row Row Row D0 D1 D2 D3 D4 D5... 10h Command

PA -> Acc ; Read status
BIT #6 set ;
JMP NZ LOOP1 ; Jmp if Busy to Loop

;i DONE!
Processor Support

- Processors with native NAND controller built-in with support for 2K page:
  - Motorola i.MX21 and i.MX31 and others
  - TI Omap 2420 and 2430 and others
  - Other vendors are adding direct-NAND interface; check with your vendor
Native NAND Interface on Freescale i.MX21

diagram courtesy Freescale Semiconductor
Single-Level Cell (SLC) vs. Multi-Level Cell (MLC)
What is the Difference?

- SLC (single-level cell)
  - SLC stores 2 states per memory cell and allows 1 bit programmed/read per memory cell

- MLC (multi-level cell)
  - MLC NAND stores 4 states per memory cell and allows 2 bits programmed/read per memory cell
SLC vs. MLC

- SLC NAND Flash products offer higher performance and reliability; typical applications include:
  - High performance media cards
  - Solid state drives (SSDs)
  - Many embedded (NAND built inside) designs including:
    - Cell phones (for executing code); MLC will still be considered for high density storage

- Multi-level cell (MLC) NAND Flash will lead in the lowest cost for consumer applications where performance and reliability are not as important; typical applications include:
  - Media players (audio and video)
  - Cell phones (SLC will still be considered for code execution)
  - Consumer media cards (such as USB, SD/MMC, and CF cards)
### SLC Attributes

**Key attributes:**

- Single bit per cell
- Supports low voltage (1.8V); required for many mobile applications
- Offered in wide data bus (16 bits) as well as 8-bit
- Supported by all controllers because SLC generally requires only 1-bit ECC
- Higher performance
- Higher reliability

<table>
<thead>
<tr>
<th>Features</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits per cell</td>
<td>1</td>
</tr>
<tr>
<td>Voltage</td>
<td>3.3V, 1.8V</td>
</tr>
<tr>
<td>Data width (bits)</td>
<td>x8, x16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Architecture</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of planes</td>
<td>1 or 2</td>
</tr>
<tr>
<td>Page size</td>
<td>2K or 4K bytes</td>
</tr>
<tr>
<td>Pages per block</td>
<td>64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reliability</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NOP (partial-page programming)</td>
<td>4</td>
</tr>
<tr>
<td>ECC (per 512 bytes)</td>
<td>1</td>
</tr>
<tr>
<td>Endurance (ERASE/PROGRAM cycles)</td>
<td>~100K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Array Operations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>†R (Max)</td>
<td>25μs</td>
</tr>
<tr>
<td>†PROG (Typ)</td>
<td>200–300μs</td>
</tr>
<tr>
<td>†BERS (Typ)</td>
<td>1.5–2ms</td>
</tr>
</tbody>
</table>
### MLC Attributes

#### Key attributes:

- Two bits per cell; twice the density of similar SLC device
- Offered only in 3.3V
- Offered only in x8 data bus
- Supported only by controllers that include 4-bit (or more) ECC
- Compared to SLC NAND:
  - Lower performance
  - Lower reliability
  - Lower price

#### Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits per cell</td>
<td>2</td>
</tr>
<tr>
<td>Voltage</td>
<td>3.3V</td>
</tr>
<tr>
<td>Data width (bits)</td>
<td>x8</td>
</tr>
</tbody>
</table>

#### Architecture

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of planes</td>
<td>2</td>
</tr>
<tr>
<td>Page size</td>
<td>2K or 4K bytes</td>
</tr>
<tr>
<td>Pages per block</td>
<td>128</td>
</tr>
</tbody>
</table>

#### Reliability

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOP (partial-page programming)</td>
<td>1</td>
</tr>
<tr>
<td>ECC (per 512 bytes)</td>
<td>4+</td>
</tr>
<tr>
<td>Endurance (ERASE/PROGRAM cycles)</td>
<td>~10K</td>
</tr>
</tbody>
</table>

#### Array Operations

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tR (Max)</td>
<td>50µs</td>
</tr>
<tr>
<td>tPROG (Typ)</td>
<td>600–900µs</td>
</tr>
<tr>
<td>tBERS (Typ)</td>
<td>3ms</td>
</tr>
</tbody>
</table>
## SLC vs. MLC

<table>
<thead>
<tr>
<th>Feature</th>
<th>SLC</th>
<th>MLC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Features</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits per cell</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Voltage</td>
<td>3.3V, 1.8V</td>
<td>3.3V</td>
</tr>
<tr>
<td>Data width (bits)</td>
<td>x8, x16</td>
<td>x8</td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of planes</td>
<td>1 or 2</td>
<td>2</td>
</tr>
<tr>
<td>Page size</td>
<td>2KB or 4KB</td>
<td>2KB or 4KB</td>
</tr>
<tr>
<td>Pages per block</td>
<td>64</td>
<td>128</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOP (partial-page programming)</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>ECC (per 528 bytes)</td>
<td>1</td>
<td>4+</td>
</tr>
<tr>
<td>Endurance (ERASE/PROGRAM cycles)</td>
<td>~100K</td>
<td>~10K</td>
</tr>
<tr>
<td><strong>Array Operations</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tR (Max)</td>
<td>25µs</td>
<td>50µs</td>
</tr>
<tr>
<td>tPROG (Typ)</td>
<td>200–300µs</td>
<td>600–900µs</td>
</tr>
<tr>
<td>tBERS (Typ)</td>
<td>1.5–2ms</td>
<td>3ms</td>
</tr>
</tbody>
</table>

- SLC is typically offered in lower voltage and wider busses.
- MLC density is 2 times that of similar SLC.
- SLC requires less ECC.
- SLC reliability is 10 times better!
- SLC performance is ~3 times better.
(72nm SLC) 4Gb Performance

Symbol | Time | Units
---|---|---
tR | 20 | us
tDCBSYR1 | 3 | us
tDCBSYR2 | 3 | us
tRC | 25 | ns
tRC (C) | 25 | ns
tRC | | ns
tPROG | 220 | us
tCBSY | 3 | us
tDBSY | 0.5 | us
tWC | 25 | ns
tWC (C) | 25 | ns
tWC | | ns
PS | 2112 | Byte
NP | 64 | Pages
(72nm MLC) 8Gb Performance

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Time (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tR</td>
<td>50</td>
</tr>
<tr>
<td>tDCBSYR1</td>
<td>7</td>
</tr>
<tr>
<td>tDCBSYR2</td>
<td>7</td>
</tr>
<tr>
<td>tRC</td>
<td>25</td>
</tr>
<tr>
<td>tRC (C)</td>
<td>25</td>
</tr>
<tr>
<td>tRC</td>
<td>25</td>
</tr>
<tr>
<td>tPROG</td>
<td>650</td>
</tr>
<tr>
<td>tCBSY</td>
<td>30</td>
</tr>
<tr>
<td>tDBSY</td>
<td>0.5</td>
</tr>
<tr>
<td>tWC</td>
<td>25</td>
</tr>
<tr>
<td>tWC (C)</td>
<td>25</td>
</tr>
<tr>
<td>tWC</td>
<td>25</td>
</tr>
<tr>
<td>PS</td>
<td>2112</td>
</tr>
<tr>
<td>NP</td>
<td>128</td>
</tr>
</tbody>
</table>

Santa Clara, CA  USA
August 22–24, 2008
While it is possible to implement 1-bit correct (Hamming code) in software, it generally does not provide a high performance solution.

Many microprocessors include NAND controllers that support 1-bit ECC.

Some newer processors are looking to include 4-bit ECC (or more) in their on-chip NAND controllers.
SLC vs. MLC Conclusions

- MLC will always provide the lowest cost per bit
- SLC will always provide the highest performance
- SLC will always provide the highest reliability
- Choose the right NAND device for the application
All NAND Flash Devices Are Not Created Equal

- Differences include:
  - Cell types
  - Architecture
  - Performance
  - Timing parameters
  - Command set

- Open NAND Flash Interface (ONFI) drives a standard interface
Two-Plane Features

- Device is divided into two physical planes, odd/even blocks
- Users have the ability to:
  - Concurrently access two pages for read
  - Erase two blocks concurrently
  - Program two pages concurrently
- The page addresses of blocks from both planes must be the same during two-plane READ/PROGRAM/ERASE operations
4Gb, Two-Plane, 2K-Page SLC NAND Architecture

- 2,112 bytes
- 1 page = (2K + 64 bytes)
- 1 block = (2K + 64) bytes x 64 pages
  = (128K + 4K) bytes
- 1 plane = (128K + 4K) bytes x 2,048 blocks
  = 2,112Mb
- 1 device = 2,112Mb x 2 planes
  = 4,224Mb

2,048 blocks per plane
4,096 blocks per device

Plane of even-numbered blocks
(0, 2, 4, 6, ..., 4092, 4094)

Plane of odd-numbered blocks
(1, 3, 4, 7, ..., 4093, 4095)
### Micron (72nm SLC) 4Gb die 2K Page Performance

**NAND Operation**

<table>
<thead>
<tr>
<th>Operation</th>
<th>MB/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Read</td>
<td>28.94</td>
</tr>
<tr>
<td>Cache Read</td>
<td>37.62</td>
</tr>
<tr>
<td>2Plane Page Read</td>
<td>33.50</td>
</tr>
<tr>
<td>Page Program</td>
<td>7.74</td>
</tr>
<tr>
<td>Program Pge Cache</td>
<td>9.56</td>
</tr>
<tr>
<td>2Plane Program Page</td>
<td>12.94</td>
</tr>
<tr>
<td>2Plane Program Page Cache Mode</td>
<td>19.05</td>
</tr>
</tbody>
</table>

### Symbols and Units

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Time</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>tR</td>
<td>20</td>
<td>us</td>
</tr>
<tr>
<td>tDCBSYR1</td>
<td>3</td>
<td>us</td>
</tr>
<tr>
<td>tDCBSYR2</td>
<td>3</td>
<td>us</td>
</tr>
<tr>
<td>tRC</td>
<td>25</td>
<td>ns</td>
</tr>
<tr>
<td>tRC (C)</td>
<td>25</td>
<td>ns</td>
</tr>
<tr>
<td>tRC</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tPROG</td>
<td>220</td>
<td>us</td>
</tr>
<tr>
<td>tCBSY</td>
<td>3</td>
<td>us</td>
</tr>
<tr>
<td>tDBSY</td>
<td>0.5</td>
<td>us</td>
</tr>
<tr>
<td>tWC</td>
<td>25</td>
<td>ns</td>
</tr>
<tr>
<td>tWC (C)</td>
<td>25</td>
<td>ns</td>
</tr>
<tr>
<td>tWC</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>PS</td>
<td>2112</td>
<td>Byte</td>
</tr>
<tr>
<td>NP</td>
<td>64</td>
<td>Pages</td>
</tr>
</tbody>
</table>
8Gb, Two-Plane, 2K-Page MLC NAND Architecture

1 page = (2K + 64 bytes)

1 block = (2K + 64) bytes x 128 pages
         = (256K + 8K) bytes

1 plane = (256K + 8K) bytes x 2,048 blocks
         = 4,224Mb

1 device = 4,224Mb x 2 planes
          = 8,448Mb
Micron (72nm MLC) 8Gb die 2K Page Performance

NAND Operation

Symbol | Time | Units
---|---|---
tR | 50 | us
tDCBSYR1 | 7 | us
tDCBSYR2 | 7 | us
tRC | 25 | ns
tRC (C) | 25 | ns
tRC | 25 | ns
tPROG | 650 | us
tCBSY | 30 | us
tDBSY | 0.5 | us
tWC | 25 | ns
tWC (C) | 25 | ns
tWC | 25 | ns
PS | 2112 | Byte
NP | 128 | Pages
16Gb, Two-Plane, 4K-Page MLC NAND Architecture

- **Cache Register**: 4,096 bytes, 218 bytes
- **Data Register**: 4,096 bytes, 218 bytes

2,048 blocks per plane
4,096 blocks per device

- **1 block**
- **1 block**

1 page = (4K + 218 bytes)
1 block = (4K + 218) bytes x 128 pages
= (512K + 27K) bytes
1 plane = (512K + 27K) bytes x 2,048 blocks
= 8,628Mb
1 device = 8,628Mb x 2 planes
= 17,256Mb

Plane of even-numbered blocks (0, 2, 4, 6, ..., 4,092, 4,094)
Plane of odd-numbered blocks (1, 3, 5, 7, ..., 4,093, 4,095)
Two-Plane, 4K-Page MLC NAND Architecture

Micron (55nm MLC) 16Gb die 4K Page Performance

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Time</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>tR</td>
<td>50</td>
<td>us</td>
</tr>
<tr>
<td>tDCBSYR1</td>
<td>7</td>
<td>us</td>
</tr>
<tr>
<td>tDCBSYR2</td>
<td>7</td>
<td>us</td>
</tr>
<tr>
<td>tRC</td>
<td>25</td>
<td>ns</td>
</tr>
<tr>
<td>tRC (C)</td>
<td>25</td>
<td>ns</td>
</tr>
<tr>
<td>tRC</td>
<td>50</td>
<td>ns</td>
</tr>
<tr>
<td>tPROG</td>
<td>900</td>
<td>us</td>
</tr>
<tr>
<td>tCBSY</td>
<td>3</td>
<td>us</td>
</tr>
<tr>
<td>tDBSY</td>
<td>0.5</td>
<td>us</td>
</tr>
<tr>
<td>tWC</td>
<td>25</td>
<td>ns</td>
</tr>
<tr>
<td>tWC (C)</td>
<td>35</td>
<td>ns</td>
</tr>
<tr>
<td>tWC</td>
<td>45</td>
<td>ns</td>
</tr>
<tr>
<td>PS</td>
<td>4314</td>
<td>Byte</td>
</tr>
<tr>
<td>NP</td>
<td>128</td>
<td>Pages</td>
</tr>
</tbody>
</table>

Santa Clara, CA  USA
August 22–24, 2008
Performance Bottlenecks
Read Throughput Limitations in NAND Today

- Read throughput limited by I/O frequency
- I/O time for NAND page ($t_{RC} = 20\text{ns}$)
  - 2K page: 42µs
  - 4K page: 86µs
- NAND array read transfer time
  - SLC: $t_R$ time is normally 20–25µs MAX
  - MLC: $t_R$ time is normally 50µs MAX
- Today for SLC NAND, the I/O time is 2–4x the array transfer time
- I/O performance must be less than or equal to array performance for maximum sustained read throughput
I/O Throughput Cannot Scale

- NAND Flash interface is asynchronous
- NAND timing parameters cannot scale indefinitely to faster speeds
- As tRC decreases, it becomes difficult for controllers to latch data output from the NAND
- As tWC for data input decreases, time to process command and address cycles does not decrease

<table>
<thead>
<tr>
<th>I/O Speed</th>
<th>40% Faster</th>
<th>17% Faster</th>
<th>25% Faster</th>
</tr>
</thead>
<tbody>
<tr>
<td>50ns I/O</td>
<td>30ns I/O</td>
<td>25ns I/O</td>
<td>20ns I/O</td>
</tr>
</tbody>
</table>
NAND array reads are parallel and very fast

- Read array bandwidth is greater than 330 MB/s (8KB read in 25µs)

Interface speed is the limiting factor
- Read bus bandwidth is only 40 MB/s (25µs clock)
SLC Program Array/Bus Performance

Interface speed is no longer the limiting factor

- Bus bandwidth is 40 MB/s (25µs)

Program performance is not so impressive

- Array program bandwidth is 33 MB/s (8KB programmed in 250µs)
ONFI and High-Speed NAND
Introduction
What is ONFI

- ONFI = Open NAND Flash Interface
- Includes NAND vendors, enablers, and customers
- Purpose is to standardize the NAND Flash interface
  - Packages
  - Timing parameters
  - Addressing
  - Command set
  - Device behavior
- Benefits
  - NAND devices self-describe their capabilities to controllers
  - Reduces time to qualify NAND devices at enablers and OEMs
NAND Flash Inconsistencies Without a Standard

- Device identification using read ID
- Array architecture and addressing
- Command set
- Timing parameters
- ECC and endurance
- Factory-marked bad blocks
- Device behavior and status
ONFI Technical Philosophy

- ONFI shall ensure no preassociation with NAND Flash at host design is required
  - Flash must self-describe features, capabilities, timings, etc., through a parameter page
  - Features that cannot be self-described in a parameter page (like number of CE#) shall be host discoverable
- ONFI should leverage existing Flash behavior to the extent possible
  - Intent is to enable orderly and TTM transition, so highly divergent behavior from existing NAND undesired
  - Where prudent for longevity or capability need, existing Flash behavior shall be modified or expanded
- ONFI needs to enable future innovation
The NAND Interface Today

- ONFI 1.0 has standardized today’s NAND interface
  - Consistent and easier for controller designers to identify and use NAND features
- ONFI 1.0 introduced timing mode 5 for faster I/O throughput
  - New standard for NAND interface performance
  - tRC / tWC = 20ns
Goals of a High-Speed NAND Interface (ONFI 2.0)

- Keep transition to high-speed interface simple
  - Keep and/or redefine original NAND signals to provide high-speed signaling without disrupting the NAND protocol and command set
  - Provide backward compatibility to asynchronous NAND interface to make device identification simple

- Increase I/O throughput with room to grow
  - Remove tRC latching limitation by adding a bidirectional source-synchronous strobe (DQS)
  - Remove tWC command and address cycle limitation by decoupling command and address processing from the data input rate

- Ensure a graceful transition from standard NAND to high-speed NAND
A fast NAND Flash interface is possible by:
- Adding bidirectional source-synchronous DQS
- Providing scalable DDR data I/O interface
- Optimizing the signaling to allow enough time to process command and address cycles
- Minimizing NAND pin capacitance

A scalable interface is needed for more than read I/O throughput. As more NAND devices are added to the bus, it is possible for even slower MLC devices to max the I/O bus bandwidth.
## Key Feature Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Standard NAND</th>
<th>HS-NAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Standard” asynchronous interface</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Synchronous interface</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>NAND command protocol</td>
<td>Standard</td>
<td>Standard</td>
</tr>
<tr>
<td>$t_{RC}$</td>
<td>$\geq 25\text{ns (SDR)}$</td>
<td>6ns (DDR)</td>
</tr>
<tr>
<td>$t_{WC}$</td>
<td>$\geq 25\text{ns (SDR)}$</td>
<td>6ns (DDR)</td>
</tr>
<tr>
<td>Standardized</td>
<td>ONFI 1.0</td>
<td>ONFI 2.0</td>
</tr>
<tr>
<td>Scalable to higher performance</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Error correction requirements</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Page size</td>
<td>2KB + 64B</td>
<td>4KB +224B</td>
</tr>
<tr>
<td>Block size</td>
<td>64 pages</td>
<td>128 pages</td>
</tr>
<tr>
<td>Cache mode</td>
<td>Some</td>
<td>Yes</td>
</tr>
<tr>
<td>$V_{IL}/V_{IH}$ and $V_{OL}/V_{OH}$</td>
<td>CMOS</td>
<td>CMOS</td>
</tr>
<tr>
<td>$V_{CCq}$</td>
<td>3.3V</td>
<td>1.7V to 1.95V</td>
</tr>
<tr>
<td>$V_{CC}$</td>
<td>3.3V</td>
<td>2.7V to 3.6V</td>
</tr>
<tr>
<td>Parameter page</td>
<td>Some</td>
<td>Yes</td>
</tr>
<tr>
<td>Package</td>
<td>TSOP</td>
<td>BGA</td>
</tr>
</tbody>
</table>

A natural extension to standard NAND
Backward-Compatible ONFI 2.0 Interface

- High-speed-capable NAND Flash devices power on using the asynchronous interface for backward compatibility
- Set features enable source-synchronous interface
- WE# becomes a fast CLK
- RE# handles data direction by becoming W/R# (Write/Read#)
- I/O[7:0] renamed to DQ[7:0] (name change only, functionally identical)
- DQS, a new bidirectional signal, is enabled
<table>
<thead>
<tr>
<th>Signals</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Async</td>
<td>Sync</td>
</tr>
</tbody>
</table>
| WE#     | CLK         | ▪ Free-running and used to latch command and address cycles  
|         |             | ▪ During idle CLK, may be stopped to save power |
| RE#     | W/R#        | ▪ Controls direction of DQ bus and DQS  
|         |             | ▪ W/R# = “1”: Data input  
|         |             | ▪ W/R# = “0”: Data output |
|         | DQS         | ▪ During data phase, each DQS rising and falling edge corresponds to a data byte  
|         |             | ▪ DQS is center-aligned for data input  
|         |             | ▪ DQS is edge-aligned for data output |
| ALE/CLE | ALE/CLE     | ▪ For synchronous mode:  
|         |             | ▪ ALE / CLE = “11”: Data transfer  
|         |             | ▪ ALE / CLE = “00”: Bus idle |
Source-Synchronous Data Output Signaling Example
Source-Synchronous Page Read
Example
As process geometry shrinks, it becomes more difficult for controllers to stay with 3.3V I/O
- Many applications today use 1.8V signaling
- Many high-speed interfaces today use smaller voltage swings so signals can transition faster
  - Example: Full-speed USB – 12 Mbit/s at 3.3V, High-speed USB – 480 Mbit/s at 400mV
NAND Flash today requires the array and I/O to operate at the same voltage
- Vcc = 2.7–3.6V
- Vcc = 1.7–1.95V
NAND Flash array operations perform best when Vcc > 1.8V, providing faster program, read, and erase times
By splitting the array voltage (Vcc) from the I/O voltage (VccQ), it is possible to get fast array operations and faster, lower-power I/O signaling
Potential high-speed voltage configurations
- Vcc = 2.7–3.6V, VccQ = 2.7–3.6V
- Vcc = 2.7–3.6V, VccQ = 1.7–1.95V
High-Density Scalability

- By providing multiple output drive strength settings, many NAND devices can share the I/O bus while maintaining I/O throughput.
- Example: 133 MT/s data throughput.

- 35Ω driver, 4 NAND die
- 25Ω driver, 8 NAND die
- 18Ω driver, 16 NAND die
High-Speed NAND Packaging

- High-speed-capable packages receive
  - DQS signal
  - Some Vcc changes to VccQ
  - Some Vss changes to VssQ

- The following packages will be transitioned to high-speed NAND
  - 48-pin TSOP
  - 63-ball BGA
Introducing a New BGA Package

- ONFI 2.0 will introduce a new BGA package
  - Accommodates high-speed *and* asynchronous-only NAND Flash devices
  - Dual x8 interface
  - More power/ground balls for lower noise
  - Signals arranged for excellent signal integrity
  - 1mm ball spacing for low cost PCB assembly
  - Accommodates ever-increasing NAND densities with two package outline options
High-Speed NAND Read Array/Bus Performance

Array program bandwidth is:
- 655 MB/s (16KB read in 25μs)
- 163 MB/s (4KB read in 25μs)

Interface speed is well matched
- Bus bandwidth is 200 MB/s (10ns [DDR])
High-Speed NAND Program Array/Bus Performance

Program performance is very impressive
- Array program bandwidth is 100 MB/s (16KB programmed in 160µs)

Interface speed is no longer a limitation to programming
- Bus bandwidth is 200 MB/s (10ns [DDR])
Four-Plane, 4K-Page, SLC High-Speed NAND Architecture
Four-Plane, 4K-Page SLC NAND Architecture (High-Speed NAND)

- Page Read
- Read Cache
- Multi-Plane Read
- Page Program
- Program Cache
- Multi-Plane Program
- Multi-Plane Program Cache

Graph showing raw throughput (MB/s) for different NAND operations.

Santa Clara, CA  USA
August 22–24, 2008
Micron can achieve 400 MB/s of programming performance using a single HS-NAND package (4 die total)

- Two channels, two-way interleave (100 MB/s per die)
- This provides a minimum density of 4GB
ONFI 2.0 Summary

- Fast source-synchronous interface
- Backward compatible with ONFI 1.0
  - Asynchronous interface support
  - ONFI protocol compatible
  - Self-identification of NAND features through parameter page
- Low-power DDR I/O
- Scalability for high-density applications
- New industry standard BGA package
- For more details on ONFI, visit http://www.onfi.org/
NAND Error Modes

- Program disturb
- Read disturb
- Data retention
- Endurance
NAND architecture is based on independent blocks.

Blocks are the smallest erasable units.

Pages are the smallest programmable units.
  - Partial pages can be programmed in some devices.

* Typical for 4Gb SLC
Program Disturb

- Cells not being programmed receive elevated voltage stress
- Stressed cells
  - Are always in the block being programmed
  - Either can be on pages not selected or in a selected page, but not supposed to be programmed
- Charge collects on the floating gate causing the cell to appear to be weakly programmed
- Does not damage cells; ERASE returns cells to undisturbed levels
- Disturbed bits are effectively managed with error correction codes (ECC)
- Partial-page programming accelerates disturbance

Strings being programmed are grounded; others are at 10V

Note: Circuit structures and voltages are representative only. Details vary by manufacturer and technology node.
Reducing Program Disturb

- Program pages in a block sequentially, from page 0 to page 63 (SLC) or 127 (MLC)
- Minimize partial-page programming operations (SLC)
- It is mandatory to restrict page programming to a single operation (MLC)
- Use ECC to recover from program disturb errors
Read Disturb

- Cells not being read receive elevated voltage stress
- Stressed cells are
  - Always in the block being read
  - Always on pages not being read
- Charge collects on the floating gate causing the cell to appear to be weakly programmed
- Does not damage cells; ERASE returns cells to undisturbed levels
- Disturbed bits are effectively managed with ECC

Note: Circuit structures and voltages are representative only. Details vary by manufacturer and technology node.
Reducing Read Disturb

- Rule of thumb for excessive reads per block between ERASE operations
  - SLC – 1,000,000 READ cycles
  - MLC – 100,000 READ cycles
- If possible, read equally from pages within the block
- If exceeding the rule-of-thumb cycle count, then move the block to another location and erase the original block
- Establish ECC threshold to move data
- Erase resets the READ DISTURB cycle count
- Use ECC to recover from read disturb errors
Data Retention

- Charge loss/gain occurs on the floating gate over time; device threshold voltage trends to a quiescent level.

- Cell is undamaged; block can be reliably erased and reprogrammed.

Note: Circuit structures and voltages are representative only. Details vary by manufacturer and technology node.
Improving Data Retention

- Limit PROGRAM/ERASE cycles in blocks that require long retention
- Limit READs to reduce read disturb
- Review JEDEC (JESD47) standard

<table>
<thead>
<tr>
<th>Retention Required (arbitrary time)</th>
<th>5 yr</th>
<th>2 yr</th>
<th>0.5 yr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrequently cycled blocks have longer retention</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequently cycled blocks have shorter retention</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block Cycles (arbitrary cycles)</th>
<th>10 cyc</th>
<th>1,000 cyc</th>
<th>10,000 cyc</th>
</tr>
</thead>
</table>
- PROGRAM/ERASE cycles cause charge to be trapped in the dielectric
- Causes a permanent shift in cell characteristics—not recovered by erase
- Observed as failed program or erase status
- Blocks that fail should be retired (marked as bad and no longer used)

Note: Circuit structures and voltages are representative only. Details vary by manufacturer and technology node.
Endurance Recommendations

- Always check pass/fail status (SR0) for PROGRAM and ERASE operations
  - Note: READ operations do not set SR0 to fail status
- If fail status after PROGRAM, move all block data to an available block and mark the failed block bad
- Use ECC to recover from errors
- Write data equally to all good blocks (wear-leveling)
- Protect block management/meta data in spare area with ECC
Wear-Leveling

- Wear-leveling is a plus on SLC devices where blocks can support up to 100,000 PROGRAM/ERASE cycles.
- Wear-leveling is imperative on MLC devices where blocks typically support fewer than 10,000 cycles.
- If a block was erased and reprogrammed every minute, the 10,000 cycling limit would be exceeded in just 7 days!
  \[60 \times 24 \times 7 = 10,080\]
- Rather than cycling the same block, wear-leveling involves distributing the number of blocks that are cycled.
Wear-Leveling (continued)

- An 8Gb MLC device contains 4,096 independent blocks.
- Using the previous example, if the cycles were distributed over 4,096 blocks, each block would be programmed fewer than 3 times (vs. 10,800 cycles if the same block is cycled).
- If perfect wear-leveling was performed on a 4,096-block device, a block could be erased and programmed every minute, every day for 77 years!

\[
\frac{10,000 \times 4,096}{60 \times 24} = \frac{40,960,000}{1,440} = 28,444 \text{ days} = 77.9 \text{ years}
\]

- Consider static vs. dynamic wear-leveling.
ECC Can Fix Everything (well, almost)

- Understand the target data error rate for your particular system
- Understand the use model that you intend for your system
- Design the ECC circuit to improve the raw bit error rate (BER) of the NAND Flash, under your use conditions, to meet the system’s target BER
ECC Code Selection is Becoming Even More Important

As the raw NAND Flash BER increases, it becomes more important to match the ECC to the application’s target BER.

For SLC
A code with a correction threshold of 1 is sufficient

For MLC
$t = 4$ required (as a minimum) for MLC
Another Option: e-MMC™
Embedded Memory

- The complexities of future MLC require increased attention; the ECC algorithm, for example, is becoming more and more complex, moving from 4+ bits to 8+ bits in the future.

- A managed interface addresses the complexities of current and future NAND Flash devices; this means the host does not need to know the details of NAND Flash block sizes, page sizes, planes, new features, process generation, MLC vs. SLC, wear-leveling, ECC requirements, etc.

- e-MMC™ embedded memory is the next logical step in the NAND Flash evolution for embedded applications because it turns a program/erase/read device with bad blocks and bad bits (NAND Flash) into a simple write/read memory.
Micron Solution: e-MMC Embedded Memory (Managed NAND)

- MLC NAND + MMC 4.3 controller in one device
- High-speed solution:
  - Host selectable x1, x4, and x8 I/Os
  - 52 MHz clock speed (MAX) – 416 Mb/s data rate (MAX)
- Fully backward compatible with previous MMC systems
- ECC, wear-leveling, and block management (built in)

12 x 16 x 1.3mm BGA package
NAND Flash is the lowest cost, nonvolatile memory available today

Complexities of MLC NAND require increased hardware and software design

All these complexities are addressed through the use of the controller included with eMMC embedded memory
Reference Material

- Micron presentations and webinars:
  http://NAND.com
- Micron documentation (specifications, technical notes, and FAQs):
  http://www.micron.com/products/nand/
- The Error Correcting Codes (ECC) Page:
  http://www.eccpage.com/
- Standards
  - MultiMediaCard Association (MMCA):
    http://www.mmca.org/
  - JEDEC:
    http://www.jedec.org/
  - Open NAND Flash Interface (ONFI) Workgroup:
    http://www.onfi.org/
  - SD Card Association (SDA):
    http://www.sdcard.org/
Thank You