cādence™

INVENTIVE

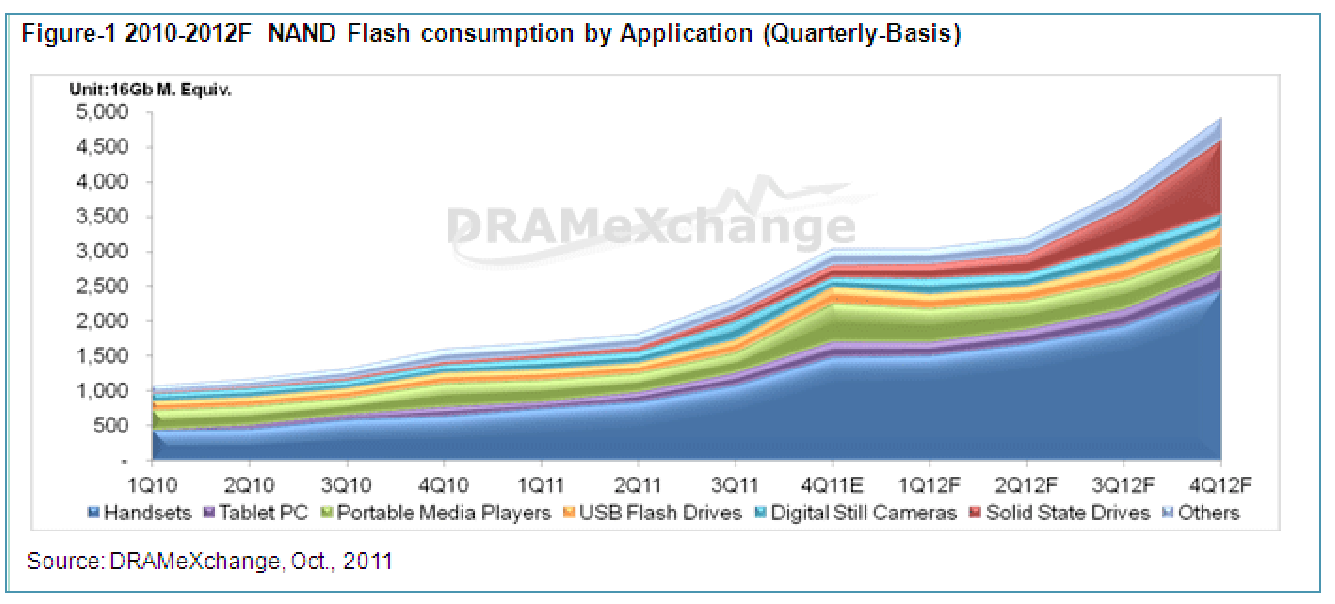
Non-Volatile Design Verification Challenges

Scott Jacobson

Agenda

- NAND Flash growth
- ECC Problems and Approaches
- ECC Trends
- Verification challenges
- Alternatives
- Future

Demand forecast by application



Growth in SSDs adding to the consumption growth of Handsets and Tablets

Demand forecast

Worldwide NAND Flash Revenue Forecast (Billions of US Dollars)

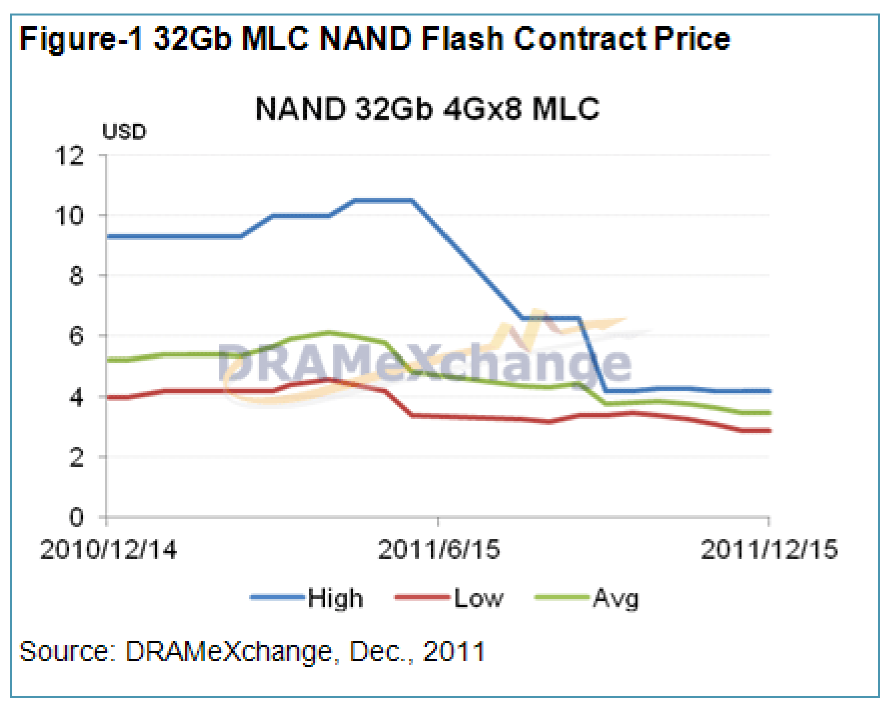
$35.0
$30.0
$25.0
$20.0
$15.0
$10.0
$5.0
$-

2011 2012 2013 2014 2015 2016

Source: IHS iSuppli Research, April 2012

NAND Flash growth trends are worldwide

Pricing Pressure

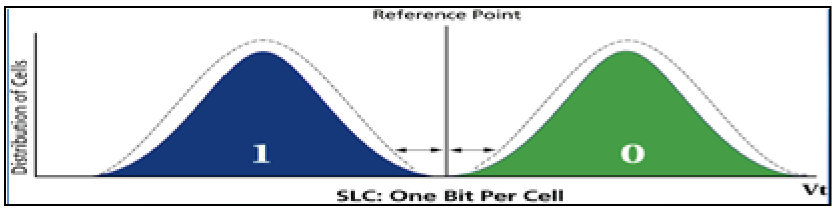


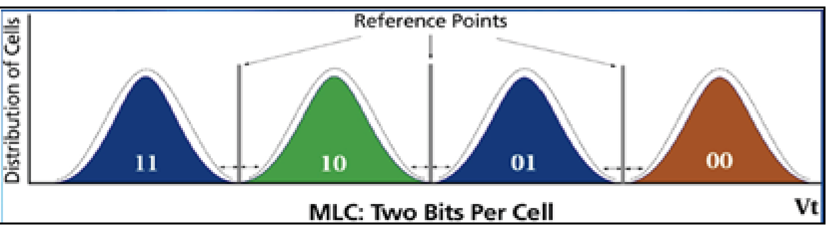
At the same time that demand is increasing, prices keep dropping

MLC versus SLC

Growth pressures for higher density and higher performance has driven the transition from SLC to MLC

This transition does not come at no cost, increased capacity also increases sensitivity to errors





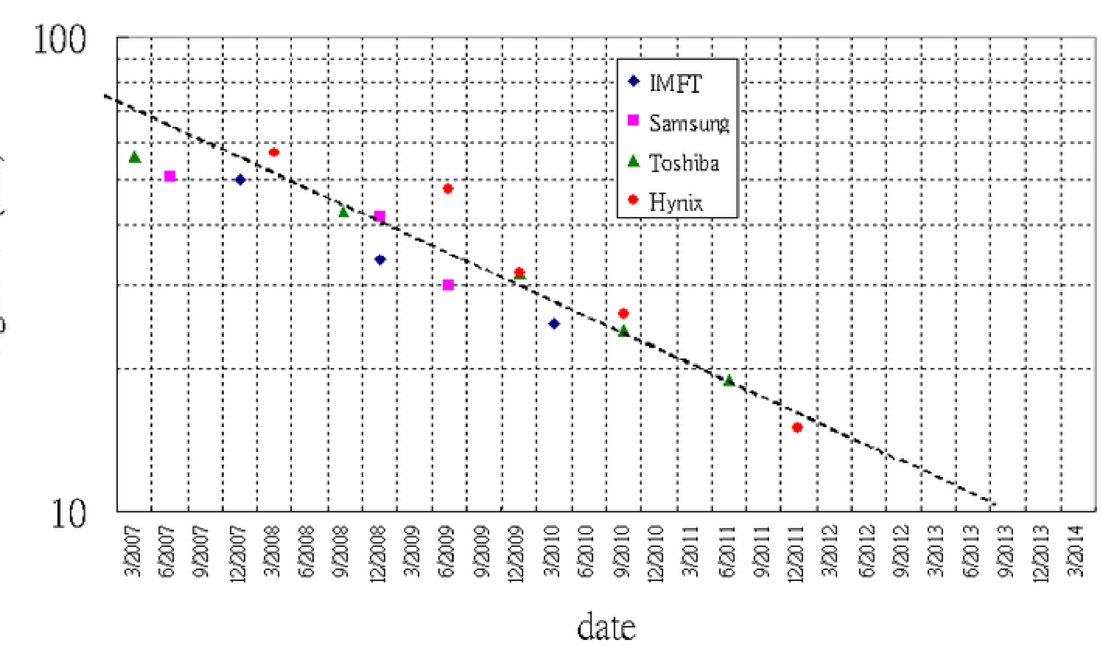
MLC –vs- SLC, cont…

Not only does the capacity increase demand higher amounts of ECC per block, but it also decreases endurance

Reliability		
	SLC	MLC
Endurance (ERASE/PROGRAM cycles)	<100000	<10000
NOP (partial page programming)	1	4
ECC (per 512 bytes)	1	4+

NAND Flash Scaling

Additionally, in order to keep up with the lowest cost manufacturing solutions, NAND Flash technology must also scale with process technology



Agenda

- NAND Flash growth
- ECC Problems and Approaches
- ECC Trends
- Verification challenges
- Alternatives
- Future

Wear Leveling

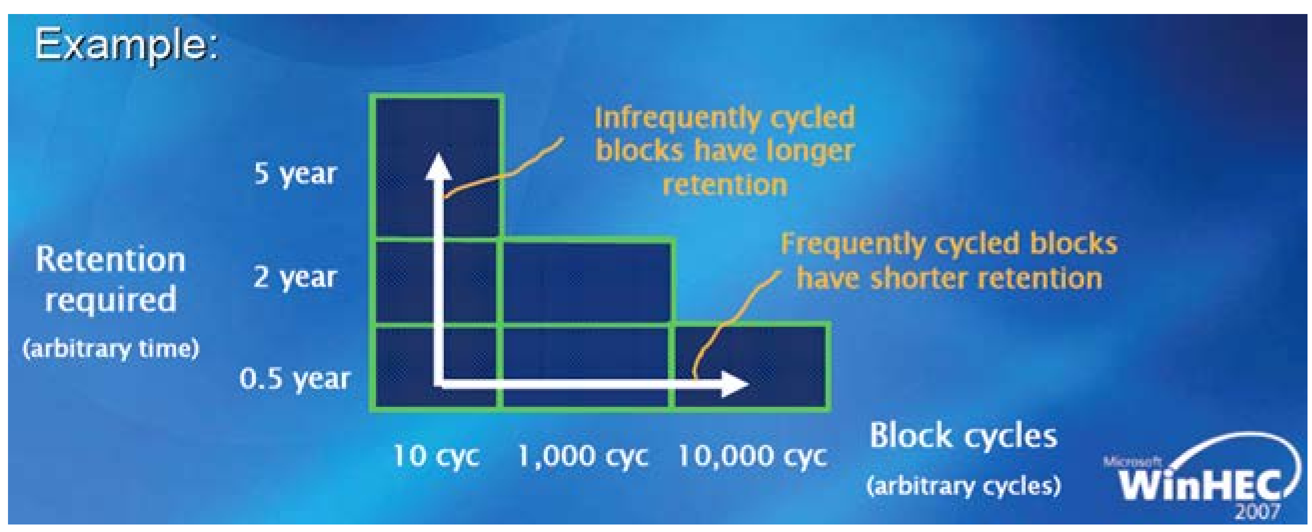
- Required on both SLC and MLC devices
  – SLC typically endures up to 100k Program/Erase Cycles
  – MLC typically endures up to 10k Program/Erase Cycles
- Distributes the write cycles more evenly across the entire FLASH device

Read Disturb

- High # of reads can cause changes in surrounding cells if not re-written
- Results in data loss
- Periodic re-write of surround cells alleviates the problem

Data Retention

- Data Retention endurance affected by program/erase cycles
  – Limit reads to reduce read disturb
  – Limit program/erase cycles in blocks that need long retention



FLASH ECC Approaches

Current ECC approaches

Traditional:

- Hamming Code
- Reed-Soloman

•Modern

- Bose, Ray-Chaudhuri, Hocquenghem (BCH)
    - Low Latency, Small gate counts, small overhead, limited ECC
- Long BCH
    - Medium Latency, Medium gate counts, small overhead, better ECC
- Low Density Parity Check (LDPC)
    - High Latency, Large gate counts, small overhead, best ECC

FMS 2010: Cadence

FLASH ECC approaches

- Hamming code
  - Typically use on SLC NAND Flash
  - Single bit error correction
  - Two bit detection
  - Typical bit-block Sizes (Block, Data)
    - (7,4) – 3 Bytes/512 Byte sector
    - (15,11)
    - (31,26)

FLASH ECC approaches

- Reed-Soloman
  - Typically used on MLC NAND Flash
    - USB
    - Memory cards
  - Typical RS code: RS (255, 223) with 8-bit symbols
    - 255 code word bytes
      - 223 data bytes
      - 32 parity bytes

FLASH ECC approaches

- BCH (Bose, Ray-Chaudhuri, Hocquenghem)
  - Typically used on MLC NAND Flash
    - SD Cards
    - SPI
    - eMMC
    - Embedded NAND
  - Multi-bit correction
  - Improved efficiency over Reed-Soloman
    - Simpler encoding/decoding techniques
    - Detects concentrated and scattered errors

Agenda

- NAND Flash growth
- ECC Problems and Approaches
- ECC Trends
- Verification challenges
- Alternatives
- Future

FLASH ECC Trends

Page size migration	1,2,4,8K soon 16K page size
Write cycles (MLC typ.)	800, 1200,1500 usec
Program erase cycles	1500, 2200,3500 usec

Table 2: Device shrinks lead to increased page size, write cycles, and program erase cycles (numbers are industry average, not manufacturer-specific)

Some applications may not see a huge number of writes compared to a cache application, where writes are the most limiting parameter.

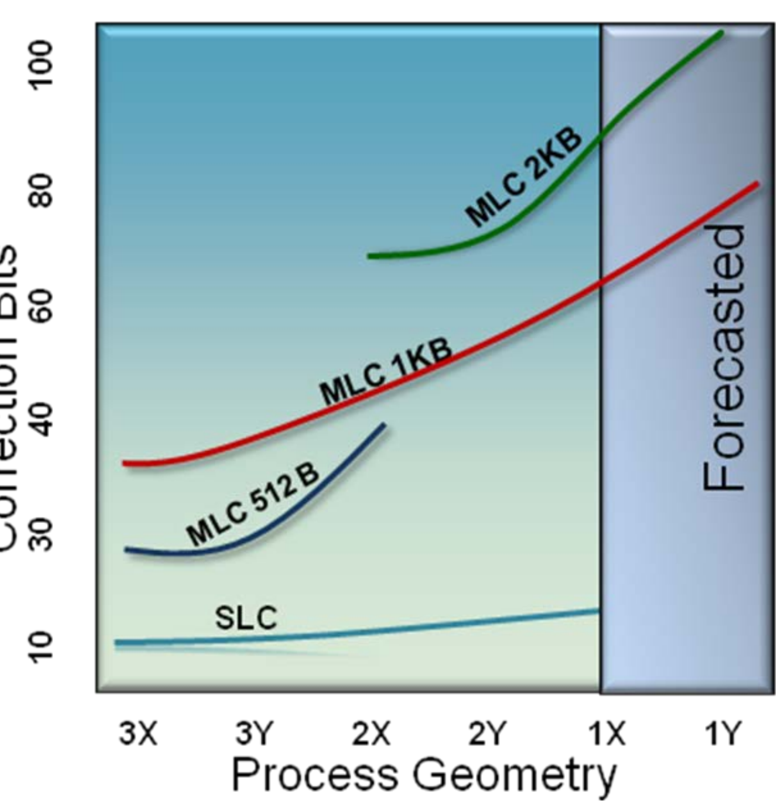
For embedded applications where booting and code are the dominating attributes, error correction coding (ECC) and known-good boot blocks are the most important features.

ECC has been increasing at each process node shrink.

ONFI 3 Enables High-Speed NAND Flash Applications
Author: Bob Pierce, Cadence Design Systems

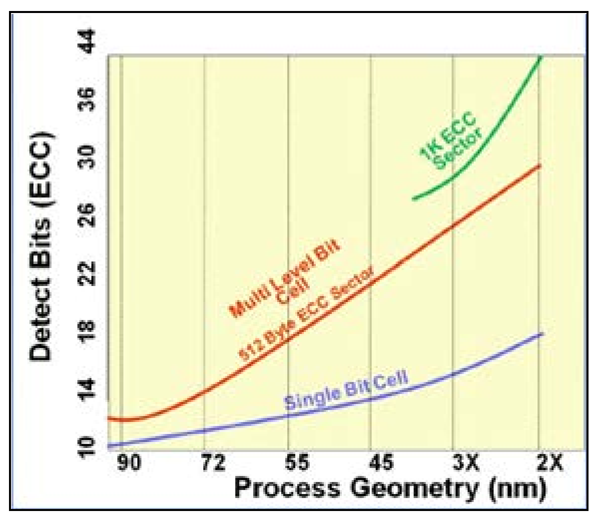
FLASH ECC Trends

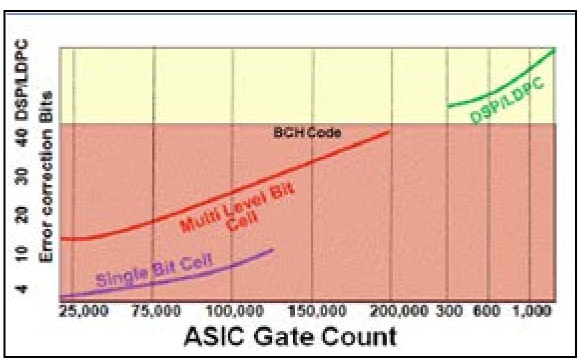
Increasing ECC demands drives the cost of increasing number of correction bits to keep in step with the larger capacities



ONFI 3 Enables High-Speed NAND Flash Applications
Author: Bob Pierce, Cadence Design Systems

ECC Flash Trends





Increasing demands for multiple bit error correction requires use of multiple ECC methods

FMS 2010: Cadence

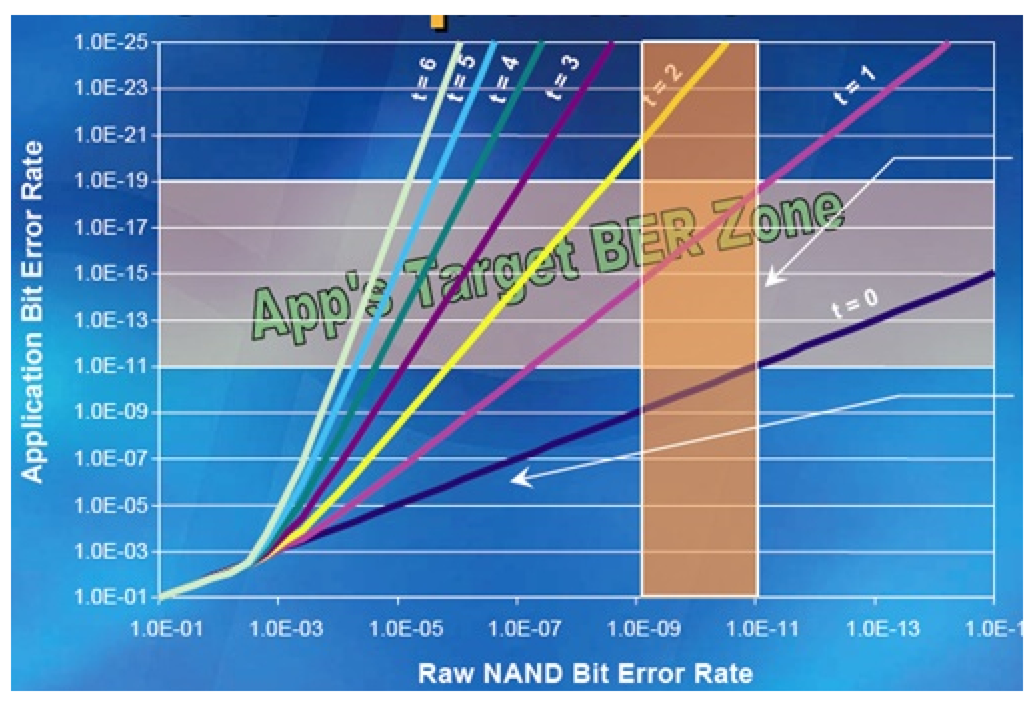
cadence™

ECC application

Different applications require different approaches to ECC

Use model for systems guide the cost/benefit tradeoff of ECC

Ultimate focus is to establish acceptable Bit Error Rates (BER) based on use model and optimize ECC costs to maintain that target



WinHEC 2007

Agenda

- NAND Flash growth
- ECC Problems and Approaches
- ECC Trends
- Verification challenges
- Alternatives
- Future

Memory Verification Challenges

- To effectively verify ECC performance in simulation, one needs to be able to
  - Inject errors into memories during simulation
  - Set assertions
    - Memory
    - Data
    - Parity Checking
  - Explore memory space
    - Width expansion
    - Depth expansion
    - Interleaving
    - Address scrambling
  - Internal memory register access

Memory Model Verification Features

- Error injection and Fault modeling for memory and ports
  - $mmerrinject *interval *data bit *percentage of errors

```
$mmerrinject(m_id, "-seed 12 -reads 5 10 -bits 1 2 4 –

  percent 80 15 5");
```

  - $mmfault, $mmsigfault *stuck-at *transition *coupling

```
mmfault(<instance id>,"<type> ",<addr>,<bit>,<value>,<slave addr>,<slave bit>);

    // Stick bit 5 of address=0x40 to "0"
     err_id = $mmfault(m_id, "stuck-at", 'h40, 'h5, 'h0);

   // Make bit 2 of address=0x20 unable to transition to a '1'
    err_id = $mmfault(m_id, "transition", 'h20, 'h2, 'h1);
```

Memory Model Verification Features

- Setting assertions on memory access, data access and parity checking
  - $mmassert_access

mmassert_access(<instance id>,“<access>”,“<action>”,<start address>,<end address>,[<address increment>]);

*** Read - a read of a memory location.**

*** Write - a write to a memory location.**

*** ReadorWrite - either a read or a write**

*** ReadnoWrite - a read of a memory location without a previous write to that location.**

*** ReadRead - two consecutive reads to the same memory location.**

*** WriteWrite - two consecutive writes to the same memory location.**

```
//  stop the  simulation whenever locations 0x0 through 0x100 are written:
  $mmassert_access(m_id,”write”,”break”,’h0,’h100,’h1);
```

Memory Model Verification Features

- Creating system memory for width expansion, depth expansion, interleaving, address scrambling
  - $mmcreatesysmem ,
  - $mmaddtosysmem
  - $mmaddressmap

Accessing internal registers of memory

- Internal registers of the memory gets automatically instantiated when you instantiate a type of memory

```
# *Denali* Class: ddr3sdram  Instance: "testbench.model" Size: 524288Kx16

# *Denali* Class: internal  Instance: "testbench.model(initStatus_registers)" Size: 1x32

# *Denali* Class: internal  Instance: "testbench.model(mode_registers)" Size: 4x32

# *Denali* Class: internal  Instance: "testbench.model(bank_status_registers)" Size: 16x32

# *Denali* Class: internal  Instance: "testbench.model(model_status_registers)" Size: 1x32

# *Denali* Class: internal  Instance: "testbench.model(multi_purpose_register)" Size: 8x1

# *Denali* Class: internal  Instance: "testbench.model(den)" Size: 3x32
```

- The internal registers can be accessed independently as they have unique instance id’s

```
integer memId, bankId, modeId;

memId  = $mminstanceid("testbench.model");
modeId = $mminstanceid("testbench.model(mode_registers)");
bankId = $mminstanceid("testbench.model(bank_status_registers)”);
```

Agenda

- NAND Flash growth
- ECC Problems and Approaches
- ECC Trends
- Verification challenges
- Alternatives
- Future

What’s next?

- Managed NAND Solutions
  - Vendor and technology independence
    - ECC
    - Partial page program operations
    - Commands and interfaces
  - eMMC 4.5
  - UFS

eMMC

What is eMMC?

Direct NAND Interface

NAND

Processor

Block Management

ECC

Driver

NAND bus

NAND

Managed NAND

eMMC

Processor

Driver

MMC bus

MMC Controller

Block Management

ECC

NAND

Managed NAND

Microsoft WinHEC 2007

UFS..

Universal Flash Storage (UFS) System Architecture

Application

Mass Storage

SCSI Upper Layer

SCSI Mid Layer

SCSI Low Layer (UFS Driver)

Software

UFS Host

MMIO

UPIU

UFS Host Controller

DME

CPORT

MIPI Unipro

MIPI M-PHY

Hardware

UFS Memory Device

MIPI M-PHY

MIPI Unipro

UFS Device Controller
UFS Memory

Agenda

- NAND Flash growth
- ECC Problems and Approaches
- ECC Trends
- Verification challenges
- Alternatives
- Future

Trends

- ECC approaches continue to be more complex and must be used judiciously based on application targets
- High IOPS application requirements continue the drive for higher transfer rates
- Managed NAND solutions becoming more complex in implementation but more standardized in approaches
- Don’t forget verification

cādence™

NAND Architecture

NAND Architecture

- 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

Block architecture

I/O
I/O
I/O
I/O
Control gate
String
Memory cell
Float gate
64 pages per block*
Page
16,896 bits per page*

* Typical for 4Gb SLC

Microsoft WinHEC 2007

FLASH ECC background

- NAND flash architecture was introduced by Toshiba in 1989. These memories are accessed much like block devices, such as hard disks or memory cards. Each block consists of a number of pages. The pages are typically 512[14] or 2,048 or 4,096 bytes in size. Associated with each page are a few bytes (typically 1/32 of the data size) that can be used for storage of an error correcting code (ECC) checksum.
- Typical block sizes include:
- 32 pages of 512+16 bytes each for a block size of 16 KB
- 64 pages of 2,048+64 bytes each for a block size of 128 KB[15]
- 64 pages of 4,096+128 bytes each for a block size of 256 KB[16]
- 128 pages of 4,096+128 bytes each for a block size of 512 KB.
- While reading and programming is performed on a page basis, erasure can only be performed on a block basis.[17] Number of Operations (NOPs) is the number of times the pages can be programmed. So far, this number for MLC flash is always one, whereas for SLC flash, it is four.[*citation needed*]
- NAND devices also require bad block management by the device driver software, or by a separate controller chip. SD cards, for example, include controller circuitry to perform bad block management and wear leveling. When a logical block is accessed by high-level software, it is mapped to a physical block by the device driver or controller. A number of blocks on the flash chip may be set aside for storing mapping tables to deal with bad blocks, or the system may simply check each block at power-up to create a bad block map in RAM. The overall memory capacity gradually shrinks as more blocks are marked as bad.
- NAND relies on ECC to compensate for bits that may spontaneously fail during normal device operation. A typical ECC will correct a one-bit error in each 2048 bits (256 bytes) using 22 bits of ECC code, or a one-bit error in each 4096 bits (512 bytes) using 24 bits of ECC code.[18] If the ECC cannot correct the error during read, it may still detect the error. When doing erase or program operations, the device can detect blocks that fail to program or erase and mark them bad. The data is then written to a different, good block, and the bad block map is updated.
- Most NAND devices are shipped from the factory with some bad blocks. These are typically marked according to a specified bad block marking strategy. By allowing some bad blocks, the manufacturers achieve far higher yields than would be possible if all blocks had to be verified good. This significantly reduces NAND flash costs and only slightly decreases the storage capacity of the parts.
- When executing software from NAND memories, virtual memory strategies are often used: memory contents must first be paged or copied into memory-mapped RAM and executed there (leading to the common combination of NAND + RAM). A memory management unit (MMU) in the system is helpful, but this can also be accomplished with overlays. For this reason, some systems will use a combination of NOR and NAND memories, where a smaller NOR memory is used as software ROM and a larger NAND memory is partitioned with a file system for use as a non-volatile data storage area.
- NAND sacrifices the random-access and execute-in-place advantages of NOR. NAND is best suited to systems requiring high capacity data storage. It offers higher densities, larger capacities, and lower cost. It has faster erases, sequential writes, and sequential reads.

cadence™

ECC Hardware Example

Flash Memory SUMMIT

Centralized ECC Hardware Architecture

BUFFER
RAM Controller
Multi-Port Arbitration Unit
HOST SYSTEM
P H Y
Host Interface Controller
DMA
CMD & DATA DMA
NAND Controller Channel 1
CRC
P H Y
1 … x NAND Devices
NAND Controller Channel 2
CRC
P H Y
1 … x NAND Devices
1 … x controller channels
NAND Controller Channel n
CRC
P H Y
1 … x NAND Devices
LDPC
BCH, ECC
Buffer
Processor
cadence