Optimizing e.MMC Memory on Intel® (Altera®) SoC FPGA Platforms

Justin Hunter, Application Engineer
Micron Technology
Embedded Business Unit
Objective

- Managed NAND devices, like e.MMC, provide not only industry standard features and characteristics but also detailed proprietary features.

- Though designed to be as drop-in ready as possible, many advantages are often under-utilized.

- With a little investment in software design, improvements can be made to:
  - optimize data throughput
  - improve data reliability
  - increase life span

- This presentation will cover a few best practice methods on how to enable these capabilities of the e.MMC memory on the Intel® (Altera®) SoC FPGA platforms.
Agenda

• Arria® 10 SoC FPGA System and e.MMC Overview
• Wrestling with e.MMC in a System
• Rules of Thumb
• Improvement Techniques
• Summary
e.MMC in Arria® 10 SoC FPGA Hard Processor System (HPS) Block Diagram

- e.MMC is used for:
  - Unified storage for boot, OS, FPGA configuration and other data
  - Lower cost/bit storage
  - ease of development converting SD Card usage → robust e.MMC
- e.MMC is useful in various industrial, automotive and consumer applications
Architecture and Interface of e.MMC

- Embedded Managed NAND system:
  - Controller
    - Core Regulator
    - Internal error correction code (ECC) engine
    - NAND Block management
  - NAND stack

- Interface with host (Logical Block Address):
  - CMD: bidirectional channel for device initialization and command transfers
  - DAT [7:0]: bidirectional I/O data signals
    - 1-bit: dat[0]
    - 4-bit: dat[3:0]
    - 8-bit: dat[7:0]
  - CLK: timing signal
    - Up to 52MHz (SDR/DDR, HS200 (SDR/DDR)
    - Additional Data Strobe (DS) for HS400

- e.MMC Power supply
  - Vcc: NAND core and I/O power supply
  - Vccq: Embedded controller core & e.MMC power supply
  - Vss, Vssq: Common ground
  - Vddi: Coupling for core regulator
Challenges of e.MMC Behavior

• What do we want from an e.MMC?
  - Faster throughput (“Performance”)
  - Better data reliability (“Data Retention”)
  - Longer life (“Endurance”)

• How can we get more out of the e.MMC?
  - Supplier improvements
    ▪ Time consuming
    ▪ Requalification complications
  - Employ simple software techniques
    ▪ Typically faster results
    ▪ Especially helpful during early design process
    ▪ Better allowance of forward-compatibility
Rules of Thumb for e.MMC

• Managed NAND is a system—it is NOT the same as discrete NAND
  - Strive to keep it orderly!
• Sequential writes are best
  - Faster throughput
  - Less “defragmentation” maintenance
• File systems are frustrating
  - File systems can complicate e.MMC functions and increase wear on the NAND
• Avoid unnecessary data removal
  - Logical erase procedures amplify actual erase cycles
• Between data retention and endurance, err on the side of data retention
  - Intact read-only data is better than no data
Software Block Diagram with e.MMC

- User Application
  - Improvement limited to user applications
- File System
  - Many file systems have configuration options
- Driver
  - Often open source and good for enabling or improving e.MMC features
- Hardware
  - Can be changed but high investment in time and money
Utilize Register Settings

• Aligning system behavior to many register setting values results in increased performance, lower write amplification (longer life), and even some data retention improvement

• Some useful values:
  - Optimal Write/Read Sizes – Performance
    ▪ Good for data transfer “chunk” size and file system block size (typically 4KB – 32KB)
  - NAND page access size (“ACC size”) – Performance, Endurance, Data Retention
    ▪ Good for data file/packet size (typically 32KB – 512KB)
  - High Capacity Write Protect/Erase – Performance, Endurance, Data Retention
    ▪ Best size for file system partitions and boot code size (512KB – 16MB)
  - Life estimate and EOL – Endurance, Performance
• Timing parameters may be equally valuable (especially u-boot)
File System Data & Partition Alignment Illustration

- Simplified management assuming a convenient register size of 10 LBA
  - Misaligned data and file system partitions result in extra performance overhead and wear on NAND

- Aligned data chunks and partition sizes reduce management requirements by the controller, increasing data throughput and reducing NAND program cycles
Example: Optimal Data Transfer Size

• In addition to software partition alignment, file system should have proper alignment with a convenient block size during the formatting procedure

• An EXT4 file system example:
  - The `mkfs -b [size]` option forces the file system to use a certain block size
  - The eCSD minimum value for "optimal write/read" is 4KB
    ▪ Best practice would be to use e.MMC register settings to determine the best block size
    ▪ "Multiblock allocation" option may have more alignment implications
Carefully consider data removal mechanisms
- File system (and OS) data removal functions may be convenient for system maintenance but can significantly—and often unnecessarily—increase NAND block erase counts affecting life span
- Operational nomenclature differences between host and e.MMC may result in decreased performance and endurance
  - For example: file system “trim” may actually result in e.MMC “secure trims”

Enable a manual e.MMC background operations (BKOPS) regimen
- Maintains optimal performance
- Refreshes weak data
- Reduction in data removal needs
- Interruptible and resumeable

Host Manual Preventative Maintenance

Rules 3, 4, & 5
Removing Data with Software Tools

Linux* “fstrim” runs a file system trim in all the card mapped area

- Some file systems do not distinguish between unmapped and unassigned resulting in excessive erases

- Actual execution depends on system settings, kernels, etc.
  - “Trim” may not always be “trim”

Besides built-in commands, add-on software tools are available like mmc-utils (left)

Some vendors may have their own add-on proprietary tools for vendor specific functions
Example: Host Manual Background Operation

- Manual BKOPS can be implemented with simple code additions or using some add-on code packages
  - Enable OS to start BKOPS when status is 1 or 2 instead of 3 (critical)
  - When usage model is known, occasional and regular instances of manual execution can be determined with little impact on life span
  - Some e.MMC may have additional user-friendly “automatic refresh” functions

Operating System example (Linux* 3.16):

<table>
<thead>
<tr>
<th>Function</th>
<th>Code pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting BKOPS</td>
<td>mmc_start_bkops</td>
</tr>
<tr>
<td>Stopping BKOPS</td>
<td>mmc_stop_bkops</td>
</tr>
<tr>
<td>High Priority Interrupt (HPI)</td>
<td>mmc_interrupt_hpi</td>
</tr>
</tbody>
</table>

Software add-on example:

```bash
# ./xxxxxxx -h
Usage: ./xxxxxxx [debug][options]... [device file]...
...  
-b --bkops          Print BKOPS info
-n --enable_bkops   Enable BKOPS
-s --start_bkops    Start BKOPS
...  
```
Health Monitoring

• Health status data provides information on remaining life expectancy
  - Standard JEDEC* health monitoring useful and applicable to all e.MMC
    ▪ Reported via several dedicated Extended CSD register values
  - Vendor specific health reports can provide considerably more detail
    ▪ May use additional dedicated Extended CSD or specialized write and read commands
  - Incorporate specialized monitoring into design as early as possible

• Vendor specific reports may provide much more detail
  - Depending on detail of vendor health reporting, data may be useful not only for lifespan estimations but also for optimizing user operations upstream
  - Vendors often “upstream” application of these features to be incorporated into new kernel versions or add-on kernel patches
    ▪ Add-on software packets may also be available from some vendors
Vendor Specific Health Monitoring

Extended CSD
• Register methods largely in place

Software Tool
• Package or patch added to existing code

JEDEC* std CMDs
• Upstream enablement

<table>
<thead>
<tr>
<th>Register Name</th>
<th>eCSD Slice</th>
<th>CMD</th>
<th>Register Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor Health Report</td>
<td>[301:270]</td>
<td>55</td>
<td>Indicates next command is application specific</td>
</tr>
<tr>
<td>Vendor Specific Fields</td>
<td>[127:64]</td>
<td>56</td>
<td>General purpose single block data write/read from non-user area</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Health Report Method</th>
<th>Advantages</th>
<th>Required Information from Vendor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended CSD Register Values</td>
<td>Simple, uniform, relatively familiar output</td>
<td>Some code adjustment to read new indexes High risk of device migration inconsistency Limited available reporting detail</td>
</tr>
<tr>
<td>Standard JEDEC Commands</td>
<td>High &quot;Upstream&quot; inclusion Large potential for detailed reporting</td>
<td>Enablement confirmation from Vendor Detail of report can be cumbersome</td>
</tr>
<tr>
<td>Software add-ons</td>
<td>Simple implementation (&quot;turn-key&quot;) Large potential for detailed reporting Easily updateable with new features</td>
<td>Software &amp; enablement confirmation from Vendor</td>
</tr>
</tbody>
</table>

# ./xxxxxxx -[arg]
Total spare block count: 310
Used spare blocks[%]: 0.00
Total initial bad block count: 0
Total later bad block count : 0
MLC area erase count Min/Max/Ave: 5 100 30
SLC area erase count Min/Max/Ave: 30 200 80
Summary and Next Steps

• Always keep rules of thumb in mind
• Some simple, low-effort software adaptations can yield quick results
  - Use register settings in software to streamline managed NAND behavior
  - Implement manual Background Operations and execute periodically for preventative maintenance
  - Value to health monitoring is immeasurable: especially vendor proprietary functions
• Software improvements “upstream” are always in the works
  - Friendly suppliers may be able to help with code review or “upstream” actions
    ▪ Proprietary vendor commands, software tools, etc.
  - Don’t wait for systemic overhauls to save a few paddle strokes!
Additional Sources of Information

A PDF of this presentation is available from our Technical Session Catalog: [www.intel.com/idfsessionsSF](http://www.intel.com/idfsessionsSF).

Demos in the ISDF Exhibit Hall – Micron Booth, Table #9

More web based info: [e.MMC Software Documentation](http://www.intel.com), [e.MMC Technical Notes](http://www.intel.com)