IP/SOC 2005

Session : Industrial Design and Methodology

Designing a CE-ATA Verification Environment for SoC Applications

Ioannis Mavroidis, Globetech Solutions

Thessaloniki, Greece

Abstract :

In this paper we will describe a verification environment developed for the emerging CE-ATA interface. The environment is written in ***e*** and is fully compatible with Cadence's Specman Elite. As such, it can be used as a Plug-n-Play verification component into any SoC that implements a CE-ATA bus. The user has full control over every verification aspect, including actively driving generated stimuli onto the bus, or passively monitoring the bus for protocol compliance checking, and coverage collection.

1. Introduction

CE-ATA [1] [2], is an emerging disk drive interface tailored to the needs of the handheld and Consumer Electronics (CE) industries. It uses an optimized subset of the ATA command set, stripped down to its bare essentials, over the proven and established MultiMedia Card (MMC [3]) electrical interface.

Creating a Verification Environment (VE) for CE-ATA poses a number of challenges. A comprehensive VE should be easily configurable to fit the user environment, should allow full user control to every aspect of its functionality, and should provide monitoring, checking and coverage collection capabilities at all protocol layers. Additionally, since the CE-ATA interface is usually part of a bigger System-on-Chip (SoC) design, a plug-n-play ability to a bigger system-wide VE is particularly important.

In this paper, we discuss these challenges and present a CE-ATA VE that has been constructed with the above goals in mind. It is written in ***e*** as a reusable and highly configurable ***e***VC (***e*** Verification Component [4]), is fully compatible with Cadence's Specman Elite [4], and takes advantage of the powerful ***e***RM (***e*** Reuse Methodology [4]) guidelines.

2. CE-ATA Technology Overview

CE-ATA, publicly announced in September 2004 at the Intel Developer Forum, is a disk drive interface standard tailored to the needs of the handheld and Consumer Electronics (CE) industries. It facilitates the adoption of higher capacity digital storage, afforded by small form factor hard disk drives, into host systems such as cameras, MP3 players, PDAs, personal video players, cellular handsets, GPS navigation systems, automotive devices, and various new and emerging applications. The CE-ATA initiative is backed by some of the most prominent companies in these market segments.

Based on the target market, CE-ATA's goals differ from the ones of its desktop siblings, PATA and SATA (Parallel and Serial ATA). Emphasis is given on simplicity, ease of integration, power efficiency and low pin-count, instead of plain performance.

CE-ATA meets these goals by using an optimized subset of the ATA command set, stripped down to its bare essentials, over the proven and established MultiMedia Card (MMC) electrical interface. Thus, ATA commands are sent to the small form factor HDD using the MMC bus protocol.

The ATA commands used by CE-ATA are:

- **IDENTIFY_DEVICE**
  This command returns a 512-byte data structure to the host that describes device-specific information and capabilities. This information includes the Serial number, Firmware revision, Model number, CE-ATA version compliance, Sector size, and ATA signature.

- **READ_DMA_EXT**
  This is the only available ATA command to read data from the device. The command parameters are: **Sector Count**

which identifies the number of 512-byte units of data to be transferred, and **LBA** which identifies the starting logical block number for the transfer.

- **WRITE_DMA_EXT**
  This is the only available ATA command to write data to the device. The command parameters are the same as the ones used in READ_DMA_EXT.

- **STANDBY_IMMEDIATE**
  This command forces the device into a power save mode.

- **FLUSH_CACHE_EXT**
  For devices that buffer/cache written data, this command ensures buffer data is written to the device media.

In ATA, a command is executed by first delivering it to the device, and then by having both host and device execute the command specific protocol.

Assuming no PACKET command support, all commands and command parameters are delivered by writing the device Command Block registers. After writing the command parameters to the corresponding device registers, the command is launched by writing to the Command register. At this point, both the host and the device enter a command specific state machine to execute the command protocol.

For the DMA transfers (assuming Multiword DMA is used, instead of Ultra DMA) the ATA command protocol consists of the device asserting a DMA request signal when it is ready for the transfer, waiting for the host to acknowledge the DMA request and then transferring the requested data. After the DMA transfer is acknowledged and finished, the host polls the device Status register or waits for a device interrupt (depending on whether interrupts are enabled or not) until the device becomes idle.

CE-ATA uses the above ATA command delivery and protocol execution mechanisms over an MMC bus and command protocol. The MMC bus conforms to the handheld market requirements, utilizing a low pin-count and low voltage levels. It consists of 3 power supply pins and 10 data pins (contrast this to the 50-pin ATA interface):

- **CLK**
  All bus signals are synchronous to this clock signal.

- **CMD**
  This signal is bi-directional and is used for the serial transfer of both the host commands and the device responses. Both commands and responses are preceded by a start bit ('0') and succeeded by a 7-bit CRC checksum followed by an end bit ('1'). All command parameters, such as address, byte counts, etc., are transferred on this signal as part of the command.

- **DAT0-DAT7**
  These are bi-directional data lines used to transfer data from the host to the device and vice versa. Data is transferred in blocks of configurable size (default is 512-byte blocks), which are preceded by a start bit ('0') and succeeded by a 16-bit CCITT type of CRC checksum followed by an end bit ('1') in each of the data lines. When data is transferred to the device, line DAT0 is also used for flow control; the device will use this as a busy signal after every data block transmission to indicate when it is ready to accept the next data block.

After a command is driven by the host on the CMD line, the device will drive its response (if any) on the CMD line, followed by the data block transfers (if any) on the DAT0-7 lines.

CE-ATA maps all required ATA registers to the MMC register space and uses a reduced MMC command set to deliver the ATA commands of the previous section to the device. It makes use of the following MMC commands (CMD60 and CMD61 are new MMC commands defined by CE-ATA):

- **GO_IDLE_STATE (CMD0)**
  Used for software resets.

- **STOP_TRANSMISSION (CMD12)**
  Used for command abortion.

- **FAST_IO (CMD39)**
  Used for single register access (8-bit). The host will use this MMC command to access a single (MMC-mapped) ATA register, e.g. for polling during ATA Command Protocol execution.

- **RW_MULTIPLE_REGISTER (CMD60)**
  Used to access multiple registers with a single MMC command. The host will use this MMC command to deliver an ATA command by writing all appropriate (MMC-mapped) ATA registers.

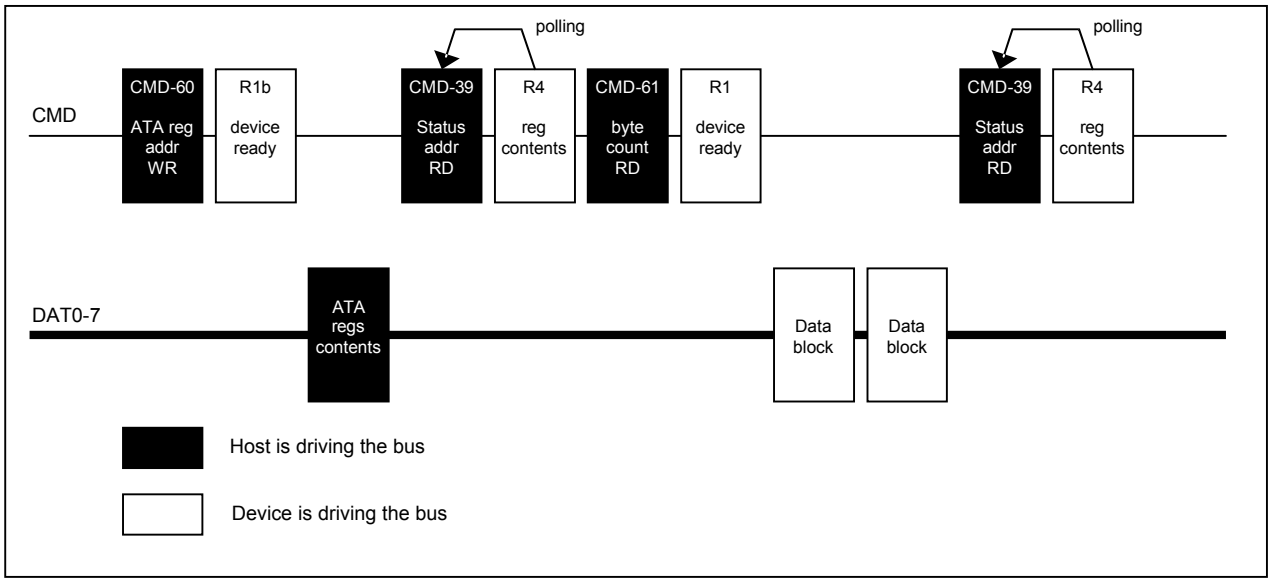

Figure 1: READ_DMA_EXT execution over MMC bus.

- **RW_MULTIPLE_BLOCK (CMD61)**
  Used to transfer data. The host will use this MMC command to transfer the data during the ATA Command Protocol execution.

CE-ATA also defines a command completion signal for use as the interrupt mechanism during the ATA Command Protocol execution (if interrupts are enabled). Instead of adding an extra interrupt line, the device interrupts the host by sending the command completion signal over the CMD line.

Figure 1 shows how a READ_DMA_EXT command would be executed over a MMC bus. The host uses CMD60 to deliver the ATA command to the device and then polls the Status register until the device is ready to accept data (DRQ set). At this point, the host uses CMD61 to initiate the data transfer. When the data has been transferred, the host again polls to determine the device status after command execution (assuming interrupts are disabled; if not, the device would use a command completion signal to interrupt the host when the command has finished).

3. Verification Goals and Challenges

Figure 2 shows the possible applications of a CE-ATA Verification Environment (VE).

In order to verify a CE-ATA compliant device, the VE should be able to emulate a CE-ATA compliant host, and vice versa. In both cases, the VE should monitor and check the CE-ATA interface for protocol compliance. The protocol checker can also be used in cases where both the host and device are part of the DUT.

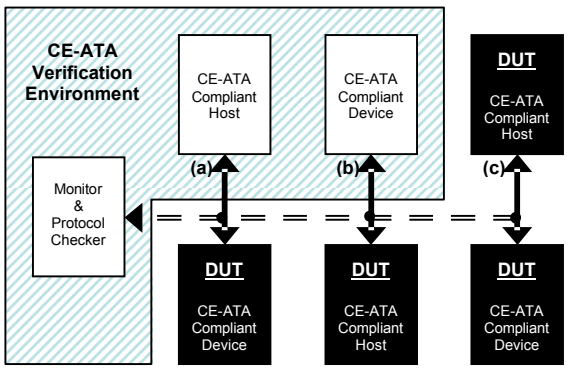

Figure 2: *Applications of a CE-ATA Verification Environment*

a) Verifying a CE-ATA compliant device
b) Verifying a CE-ATA compliant host
c) Verifying a CE-ATA Interface

Creating such a VE poses a number of challenges. These are summarized below:

- **CE-ATA protocol complexity**
  CE-ATA combines several commands and operation modes from both ATA and MMC standards. A comprehensive VE should provide the means to easily stimulate, monitor and check the CE-ATA bus for all these different modes.

- **Flexibility in user control**

The verification engineer should be able to fully control all aspects of the VE. These range from controlling whether the DUT consists of a host, a device, or both, to specifying the protocol timing parameters, and performing any optional error injection (e.g. CRC errors or device faults).

- **Layered protocol checking**
  The CE-ATA bus monitor needs to perform protocol compliance checking for the device under test (DUT) at both MMC and ATA layers. All aspects of operation need to be verified, ranging from protocol and timing violations, to transmission and data integrity errors.

- **Functional coverage analysis**
  The VE should be able to efficiently collect and report coverage data from the test-runs. Coverage analysis helps identify holes in the verification plan, eliminating which, will eventually lead to testplans that fully exercise every aspect of the DUT.

- **Verification closure metrics**
  Part of a comprehensive reusable VE is to define verification goals that can be blintly applied by the verification engineer as part of wider SoC verification goals. These goals are described in terms of coverage goals for specific items.

- **Seamless integration into SoC VE**
  Since CE-ATA designs are usually part of a bigger SoC, the verification engineer should be able to seamlessly integrate the CE-ATA VE into a bigger system-wide SoC VE. This Plug-n-Play ability plays a key role in the creation of reusable Verification IP.

4. The CE-ATA *e* Verification Component

A CE-ATA VE was constructed with the above goals in mind. It is written in ***e*** as a reusable and highly configurable ***e***VC (***e*** Verification Component), and is fully compatible with Cadence's Specman Elite. Taking advantage of the powerful ***e***RM (***e*** Reuse Methodology) guidelines, it can be put as is into a SoC-wide VE, to verify CE-ATA compliant host or device implementations.

Figure 3 shows the CE-ATA ***e***VC architecture in its two most typical applications, which are to verify a CE-ATA compliant device or host. For this reason the Agent of the CE-ATA ***e***VC can play the role of either a bus master (emulating a host that initiates activity on the bus – see Figure 3), or the role of a bus slave (emulating a device that responds to bus master requests). In both configurations, an independent Protocol Checker unit, verifies the CE-ATA bus traffic against MMC or ATA layer errors.

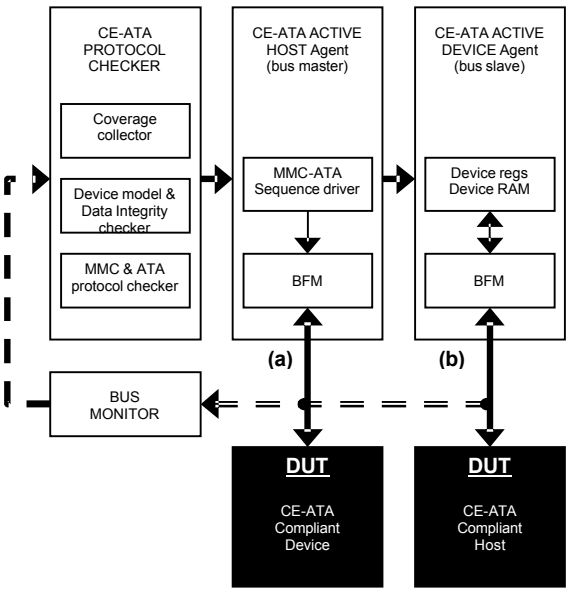

*Figure 3: Architecture of the CE-ATA **e** Verification Component (**e**VC)*

If both an ACTIVE HOST Agent and an ACTIVE DEVICE Agent are instantiated, together with the Protocol Checker, the ***e***VC can operate in a completely standalone mode, where no DUT is present. If, on the other hand, the Protocol Checker is the only unit instantiated, with no ACTIVE HOST or DEVICE Agent present, the ***e***VC can operate in a completely passive mode monitoring a CE-ATA bus.

The CE-ATA ***e***VC consists of the following components:

Bus Monitor

The Bus Monitor is a completely passive unit which sits on the CE-ATA bus, and monitors for all types of transactions including MMC Commands, MMC Responses, MMC Data Blocks, CRC Status, MMC Busy, Command Completion Signal, and Command Completion Signal Disable. It collects the data for each item in a temporary struct and, when it is fully received, will emit a corresponding event. These events will be acted upon by the HOST and

DEVICE BFMs according to the CE-ATA protocol rules, as well as the Protocol Checker.

ACTIVE HOST Agent

An ACTIVE HOST Agent, is able to fully emulate the behavior of a CE-ATA compliant host. The host acts as a bus master and generates commands on the CE-ATA bus. The user is able to fully control any aspect of the host behavior, ranging from the sequence of generated MMC and ATA commands, to the protocol timing the host should adhere to, and to any optional CRC error injection. User control is also provided at the ATA command level, where the user is able to specify specific transmission parameters for each command. The ACTIVE HOST Agent contains an MMC-ATA Sequence driver and a BFM. The user test interfaces with the Sequence driver in order to specify the sequence of commands to be executed over the bus. The BFM (Bus Functional Model) executes the host protocol to drive the bus with the specified commands.

ACTIVE DEVICE Agent

An ACTIVE DEVICE Agent, is able to fully emulate the behavior of a CE-ATA compliant device. The device acts as a bus slave and can only respond to host commands; it is not able to initiate any transactions. The user is able to fully control any aspect of the device behavior, ranging from the protocol timing the device should adhere to, to any optional CRC error and device fault injection. User control is also provided at the ATA command level, where the user is able to specify specific device reaction to each command. The ACTIVE DEVICE Agent contains models for the device registers and RAM array, as well as a BFM to execute the device protocol and drive the bus.

Protocol Checker

The role of the Protocol Checker is to decode the transactions seen on the CE-ATA bus, report any violations of the CE-ATA protocol, and gather coverage information. For this reason it models both a CE-ATA compliant host, and a CE-ATA compliant device together with its registers and storage area, thus knowing their expected states at any point during the test.

The Protocol Checker is a completely passive unit, independent of the Agents, which listens for any type of event emitted by the Bus Monitor. Whenever an event is emitted, it will be dispatched to the corresponding event handler, that will:

1. Gather coverage data for this event.
2. Check whether received event was expected at this point in the test.
3. Verify that the event adheres to all relevant timing parameters.
4. Translate it to a specific step in the CE-ATA protocol execution.

Protocol checking is performed at the following layers. The user is able to turn each layer on or off.

- **Command Parameter checking**
  This layer will report any MMC or ATA command parameter errors, including out of bounds or misaligned addresses, and illegal byte counts.

- **MMC Rules checking**
  This layer checks that all MMC Commands are correctly executed. Take for example a RW_MULTIPLE_BLOCK (CMD61) command that writes multiple blocks: this layer will verify that the MMC Command is followed by an MMC R1b Response, followed by the correct number of MMC Data Blocks, each followed by the correct type of CRC Status, and, if interrupts are enabled, followed by a Command Completion Signal.

- **MMC CRC errors checking**
  This layer will report all CRC errors seen on the bus, in any MMC command, response, or data block.

- **MMC Timing checking**
  This layer checks that the timing of all MMC transactions is observed, as defined by the $\mathbf{N_{ID}}$, $\mathbf{N_{CR}}$, $\mathbf{N_{ACIO}}$, $\mathbf{N_{WR}}$, $\mathbf{N_{CCS}}$, $\mathbf{N_{RC}}$ and $\mathbf{N_{CC}}$ timing parameters.

- **Data Integrity checking**
  This layer verifies Data Integrity of the device data seen on the bus, against the data in the device storage area model of the Protocol Checker. If a memory address was previously written with some data, the Data Integrity checking will verify that the device returns the correct data to any subsequent host reads from this memory address.

- **ATA Rules checking**
  This is the highest layer of checking and will verify that the host observes the CE-ATA protocol rules for the execution of each ATA command. For example, assuming that a Data-In command with interrupts disabled is issued, it will make sure that the host will poll the device using a FAST_IO (CMD39) command until BSY is 0, before issuing each

RW_MULTIPLE_BLOCK (CMD61) command.

Coverage Collector

The Coverage Collector makes use of Specman's Elite powerful tools for functional coverage analysis, to collect coverage information on the MMC and ATA commands, in all their different flavors, that have been exercised over the CE-ATA Bus during a set of test-runs. For example, it collects coverage information for the opcode, interrupts mode (enabled/disabled), number of CMD61's used, and MMC data block size, for all ATA commands that get executed. It also collects coverage information for the opcodes and arguments of all monitored MMC commands and responses, as well as the observed values for all protocol timing parameters.

Conclusions

CE-ATA is an emerging disk drive interface tailored to the needs of the handheld and Consumer Electronics (CE) industries. In this paper we discussed the challenges of creating a Verification Environment for the CE-ATA interface, and described an implementation that addresses these challenges in an efficient and comprehensive way.

References

[1] CE-ATA Digital Protocol, Revision 1.1, 29-September-2005. Available from http://www.ce-ata.org

[2] CE-ATA Host Design Guide, Revision 1.0, 29-September-2005. Available from http://www.ce-ata.org

[3] The MultiMedia Card System Specification, Version 4.1, April 2005. Available from http://www.mmca.org

[4] Cadence products: Specman Elite, ***e***RM (***e*** Reuse Methodology), ***e***VC (***e*** Verification Component). http://www.verisity.com/products