Reprint

# A Versatile Instrument for Analyzing and Testing the Interfaces of Peripheral Devices

*P. Savvopoulos, M. Varsamou and Th. Antonakopoulos*

The 3rd International Conference on Systems, Signals & Devices – SSD 2005

SOUSSE, TUNISIA, MARCH 2005

SSD´05
Third International Conference on Systems, Signals & Devices
March 21-24, 2005 – Sousse, Tunisia
Volume IV
Sensors, Circuits & Instrumentation Systems

# A VERSATILE INSTRUMENT FOR ANALYZING AND TESTING THE INTERFACES OF PERIPHERAL DEVICES

Panayiotis Savvopoulos[†], Maria Varsamou[‡] and Theodore Antonakopoulos[‡]

[†]Research Academic Computer Technology Institute
61 Riga Feraiou Str., 26100 Patras, Greece
Phone: +30-2610-996-489, Fax: +30-2610-996-834
E-mail: psavvop@cti.gr

[‡]Department of Electrical Engineering
University of Patras, 26500 Rio, Patras, Greece
Phone: +30-2610-996-487, Fax: +30-2610-996-834
E-mail: {varsamou, theodore}@loe.ee.upatras.gr

***Abstract*****—** *This work presents a flexible development and testing environment, created for evaluating the performance and analyzing the characteristics of interfaces used by peripheral devices. The versatility of the presented instrument is due to the integration of the Matlab/Simulink tools with a custom reconfigurable hardware platform using the PCMCIA bus of conventional laptop computers. The hardware platform uses reconfigurable circuits, acts as a host adapter of the peripheral device and transforms the test sequences of the high level software tool into interface specific bit-streams, while it simultaneously collects interface-related data for further processing.*

***Index Terms*****—** Secure Digital Memory, Performance Analysis, Consumer Peripheral Devices.

## I. Introduction

The evolution in the field of portable consumer devices, such as laptops, PDAs, MP3 players, digital cameras etc, along with new memory demanding applications have resulted to the booming of the peripheral devices market [1]. A key feature of most new emerging consumer applications is the need for reliable high rate data transfers. The peripheral devices portability and their compatibility with a large variety of consumer devices, are based on widespread standard interfaces. Representative examples of such peripheral devices are met on storage applications [2], [3].

Recently, considerable activity has been observed in the field of consumer devices development in order to satisfy the requests for faster, cost-effective and more reliable peripherals. As a consequence, there is an increasing demand for a flexible tool that can be used either for the performance analysis of various existing peripheral devices or for testing and validating new devices during development. This paper focuses on such a versatile instrument that can be used for performance analysis and testing of a wide range of interfaces for consumer peripheral devices. This task is accomplished through an integrated set-up that combines the high-performance simulation and modeling capabilities of Matlab/Simulink with a flexible and reconfigurable hardware platform that is attached to a conventional laptop computer via the PCMCIA interface.

Section II gives an overview of the development environment presented in this work. The functional description of the proposed instrument along with its architectural characteristics are discussed and analyzed in Section III. Section IV outlines the Secure Digital interface and demonstrates experimental results collected during the study of various SD memory cards.

## II. The Development Environment

The interaction of a host computer or an embedded system with its peripheral devices is based on low-level physical interfaces and a set of protocols, implemented as finite state machines (FSMs). Two kinds of FSMs are usually defined on such interfaces, one that controls the device initialization using a handshaking mechanism, while the other is used for managing the device during normal operation. The peripheral devices communicate with the host's application through a set of commands/responses that are issued by the host/peripheral on the low-level physical interface in the form of protocol-specific bit-streams. Depending on the command that the peripheral device receives, it evolves its state-machines and responds accordingly.

Performance evaluation, mainly in terms of sustained data rate, is of great importance not only for existing peripheral devices but also for new, rapidly emerging ones. Commercial host adapters allow such devices to be accessed by a host computing device. However, it is very difficult, if not impossible, to perform elaborate tests for performance analysis using these adapters, since they only allow high-level access to the device through several, operating system dependent, device I/O drivers.

This work outlines a complete development and testing environment that provides low-level control to a large variety of peripheral devices through their specific

9973-959-01-9 / © 2005 / 9885 IEEE

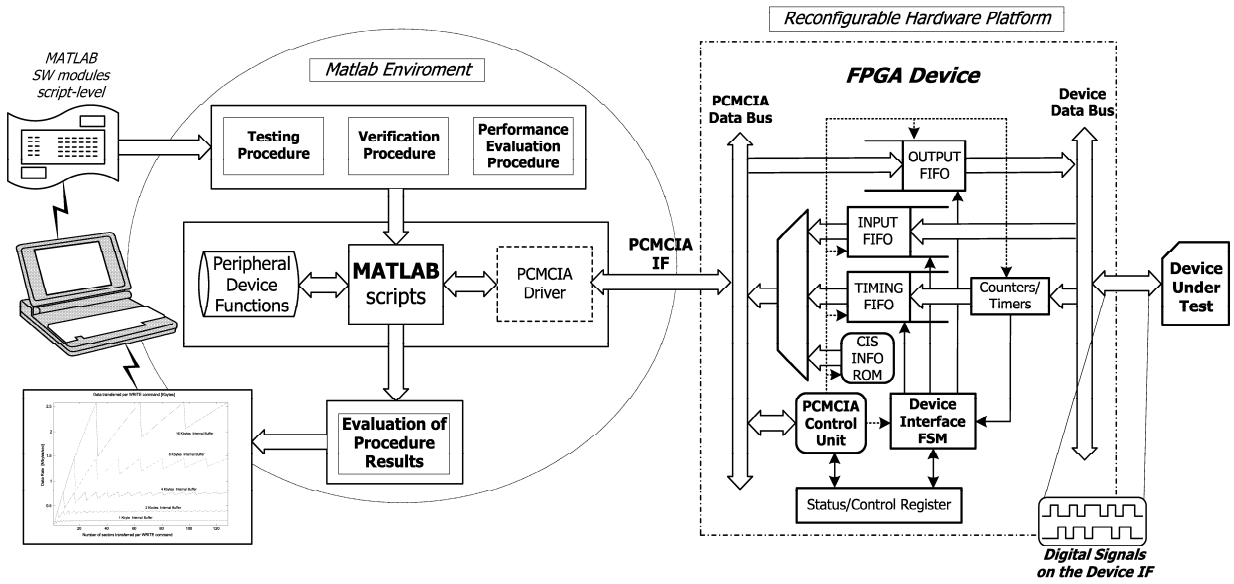


Fig. 1. The Instrument's Architecture.

interfaces, in contrast to the existing commercial host adapters. Figure 1 shows the block diagram of the proposed set-up. In particular, this environment consists of the high-level modeling tools of Matlab/Simulink that provides to the user the means to create various test scenarios and to send sets of commands to the device using a flexible hardware platform that communicates with the Matlab environment via the PCMCIA interface. The reconfigurable hardware platform, that is based on FPGA circuits, acts as a host adapter that actually issues the appropriate bit-streams to the device under test, according to the scenarios implemented in the Matlab/Simulink level. At the same time, the host adapter stores the data exchanged over the peripheral interface and records timing information regarding the concatenation of different events. All data gathered from the hardware platform together with the timing records are transferred to the software tool for further processing. The collected data are used either for testing and validation of the peripheral device functionality or for evaluating its response time and thus determining its maximum achievable data rate.

Another application of the functionality of the described set-up derives from the fact that it can be utilized as a validation tool during the prototyping of new devices that have to conform to the specifications of a given protocol. Various high-level test sequences can be generated and used to verify the proper operation of the device's state-machines and observe its behavior under erroneous circumstances. As a consequence, this systematic and methodical debugging procedure leads to error identification and correction early in the design process, thus it reduces the 'time to market' of the respective device.

## III. Functional Description

Describing in more details the internal structure of the presented instrument, the development set-up consists of the Matlab/Simulink tools and the reconfigurable hardware platform that exchanges data with the Matlab environment, through the PCMCIA interface using a custom I/O device driver and custom logic that extends the PCMCIA's accessibility.

As it is shown in Figure 1, the FPGA-based hardware platform implements the adaptation control unit required for interconnecting the peripheral device with the high-level modeling tool. The basic feature of the hardware platform's internal logic is the implementation of multiple, functionally-related FIFOs for storing and transmitting data from the Matlab to the peripheral device and vice versa. These FIFOs facilitate the interaction between the host, as it is emulated in Matlab, and the actual device. To be more specific, three types of FIFOs are used in the design, one for storing data from Matlab scripts and transmitting them properly to the used peripheral interface, one for storing the digital waveforms that appear on the interface and

transferring them to the Matlab environment and one for recording timing information about specific events such as response time, time between multiple read or write blocks etc. These FIFOs are combined with additional control logic, including timers, counters and FSMs, that allow interaction between them as the test sequences progress. As a result, testing scenarios of high complexity, that support bidirectional activities over the peripheral interface, can be executed.

Various basic functions of the peripheral device interface, such as send command/receive response, read and write data functions have been implemented as Matlab scripts. These scripts are used for creating more complex and sophisticated testing scenarios that generate a sequence of application specific commands, but still remaining independent of the characteristics of a specific physical interface. Additionally, we have to mention that the Matlab environment remains the master unit of the entire system and controls the status of the peripheral device throughout the whole procedure.

In order to perform a test scenario, various scripts are initially implemented in the Matlab environment and when they are executed, they transfer data to the specific FIFO in the hardware platform. Based on the control information also received by the Matlab scripts, the platform logic translates the stored data into signal waveforms compatible with the peripheral device's specifications. These waveforms represent either data transfers or interface specific commands. During normal operation, the peripheral device responds accordingly and evolves its internal FSM by driving the respective signal waveforms on its outputs. These signals are being recorded by specific detection logic on the platform's FPGA and are transferred to the Matlab workspace, using another FIFO. Additional internal logic in the reprogrammable circuit also involves timers and counters that acquire timing information between various interface events. This information is finally retrieved from the timing FIFO and is evaluated at the Matlab level. Such information is used in order to determine the device performance and make useful conclusions about the device functionality and its internal structure.

Following the above methodology, we are able to compare and benchmark several implementations of a peripheral interface. The presented instrument is capable of supporting different kinds of peripheral interfaces and devices with minimal changes to the adaptation control unit of the hardware platform logic along with the physical wiring. A representative example is outlined in details in the next section. This example, which is based on the Secure Digital (SD) interface, measures and compares the performance of various SD memory cards from different manufacturers.

## IV. The Secure Digital (SD) Memory Case

The above described tool has been used in order to test and validate devices supporting the SD interface [3], i.e. SD Memory cards. Several tests have been performed on commercial SD Memory Cards and various measurements have been carried out. Example figures and graphs, demonstrating the data rate achieved by various SD Cards during read and write operations, are given. The same set-up has been also used for testing SDIO cards, while it can be easily modified in order to support other similar interfaces, which are frequently used in consumer devices.

### IV-A. The SD Memory Card Architecture

A SD Memory Card is a flash-based memory card that is designed specifically to meet the capacity and performance requirements of the newly emerging audio and video consumer devices. Its communication with the host devices is based on a 9-pin interface (Clock, Command, 4xData and 3xPower lines) designed to operate in a low voltage range. The inner architecture of the card includes a set of registers, which contain several configuration parameters that control the card's operation, flash memory modules and a Card Controller that supervises its internal operation, as well as the communication between the host and the flash memory. All communication between the host device and the card is controlled by the host and is performed over the SD bus based on command and data bit-streams which are initiated by a start bit and terminated by a stop bit. Data transfers to/from the SD Memory Card are done in blocks, which are always succeeded by CRC fields. Single and multiple block operations are allowed. Various data block lengths are supported, with most common block lengths (sector size) being 512, 1024 and 2048 bytes.

The internal flash memory has various levels of organization. The read or the write operation takes place using pages, while the erase operation takes place on an erase block basis that contains multiple pages [4]. Usually the page size is a multiple of the maximum data block length supported by the SD Card. Therefore, during a SD write operation, the Card Controller stores temporarily in a local buffer so many data blocks (sectors) as needed to complete a page before transferring the data to the flash memory. Several bi-directional buffers are used, that provide a built-in 'pseudo' cache memory. Additionally, these buffers make feasible a virtual continuous write operation, allowing the host to transmit several data blocks without having to wait for each one of them to be stored in the flash memory before sending the next one. When the available buffers are full, the Card Controller asserts the first Data line

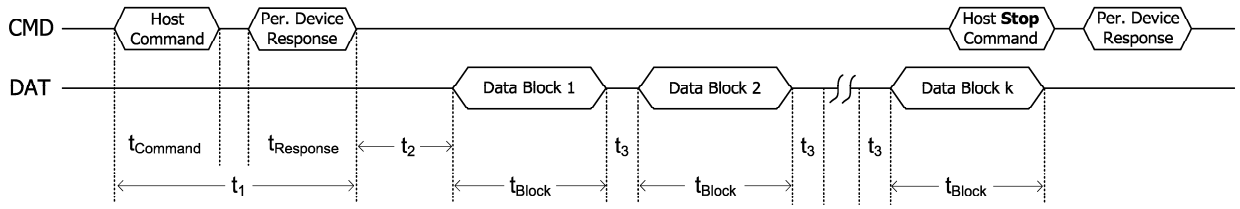


Fig. 2. A *'Read Multiple Block'* procedure at the SD memory interface.

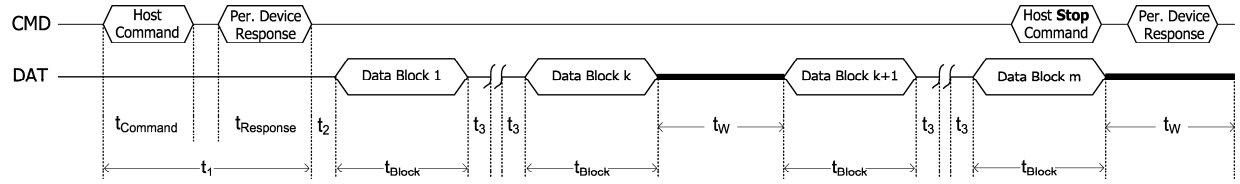


Fig. 3. A *'Write Multiple Block'* procedure at the SD memory interface.

line at low level and the host delays the transmission of the next block. The number of the buffers that the SD card uses, along with the page size of its flash memory, have a great impact on the card's performance. In practice, most SD Card manufacturers use two bi-directional buffers. Using the set-up presented in this work, we gathered several experimental data from various SD cards, which are thoroughly described in the following section.

### IV-B. Experimental Results

In this section we focus on the use of the presented instrument for performance analysis of various SD memory cards, based on sustained data rates. For this purpose we present several experimental results that demonstrate the set-up capabilities. It has to be mentioned that all the peripheral devices used during this experiment, follow the SD memory interface [2] with few variations on their properties during initialization procedure, that are irrelevant to the method followed.

In particular, using the presented set-up we emulated the performance of a SD host adapter that is capable of controling the SD memory interface bus through the Matlab environment and its running scripts, by sending any desired command to the peripheral device under test. It can also produce different clock frequencies and record every waveform that appears on the physical interface. The methodology, developed in order to perform comparison and performance evaluation regarding sustained data rates, is analyzed below.

Initially, we developed several test scenarios in the Matlab environment in the form of scripts that can read/write a large amount of data from/to the SD memory cards available. All these scripts are based on the basic functions of reading/writing sectors from/to the memory cards and on issuing commands/storing responses, as it was reported in section III. The same scripts were used for every SD card with variations only in their initialization procedure, which is necessary for being able to support data transfers. Therefore, control information is used for proper configuration of the hardware platform in terms of clock frequency, which depends on the device's state, desired initialization properties, and supported block length. This control information is also used for acquiring timing information during execution of the data transfer commands. When a Matlab script runs, the hardware platform transmits several commands to the memory card. Then, the device responses and the data exchanged over the interface along with timing information is stored in the hardware platform and is transferred to the Matlab environment when the experiment is finished.

For proper understanding of how the test sequences are performed in the physical layer and how they are related to the sustained data rate achieved on the SD memory interface, we have to define several critical time intervals that appear in the interface during execution of specific data transfer commands. The determination of these rates is feasible by performing *'Read Multiple Block'* and *'Write Multiple Block'* commands. Figures 2 and 3 present how the *'Read Multiple Block'* and *'Write Multiple Block'* commands are executed on the SD memory interface.

According to Figure 2, the exchange of a host's *'Read*

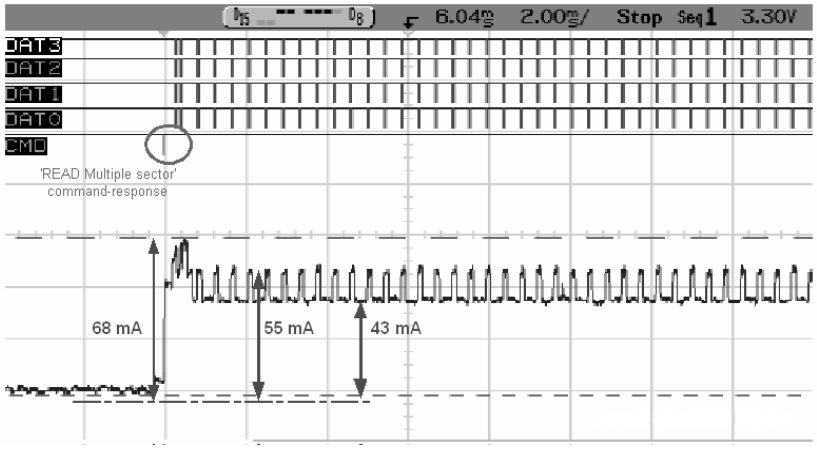


Fig. 4. An example of a *'Read Multiple Block'* command with SD Card supply current measurement.

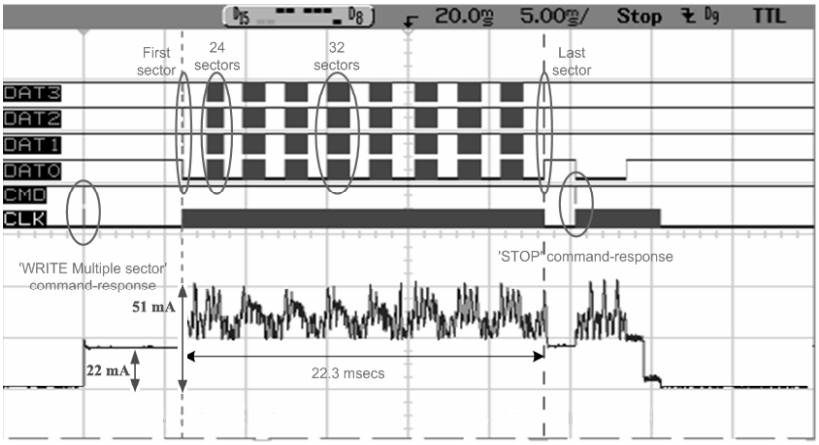


Fig. 5. An example of a *'Write Multiple Block'* command with SD Card supply current measurement.

*Multiple Block'* command and the card's response is followed by the transmission of the first data block after a card dependent time ($t_2$), which is related to the card's internal operations for activating the flash array and starting up the read data process. There is an additional guard time ($t_3$) between the transmission of successive data blocks. This process continues until a *'Stop Transmission'* command is issued by the host. The time interval $t_3$ is card dependent and for specific SD clock frequency and block length, this parameter determines the sustained data rate that can be achieved during a *'Read Multiple Block'* operation. In the following experimental results, data blocks of 512 bytes have been used.

In the case of a *'Write Multiple Block'* command, that is shown in Figure 3, we observe that the host starts transmitting data blocks after receiving the SD card's response. This procedure is terminated when the host sends a *'Stop Transmission'* command. Since the SD card uses internal buffers for temporary storage of the received data blocks, this operation continues as long as there is available temporary memory. When all internal buffers are full, the card forces the DAT0 line to low level and the host delays the transmission of the next block. When the DAT0 line is released, the host begins the transmission of the next data block. According to Figure 3, the time internal $t_w$ is card dependent and is related to the size and number of the buffers used, to the flash technology used, etc. This time determines the sustained data rate rate that can be achieved during a *'Write Multiple Block'* operation. On the other hand, the time $t_3$ is host dependent and usually, it is equal to a few clock cycles.

Figures 4 and 5 present typical examples of physical interface waveforms of a SD memory card during the execution of *'Read Multiple Block'* and *'Write Multi-*

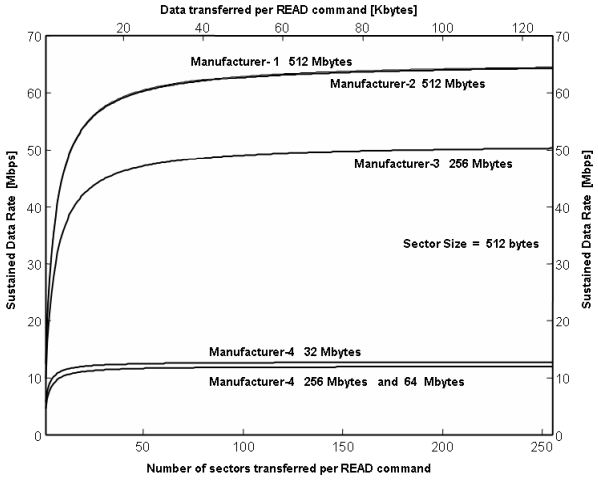


Fig. 6. Sustained Read Data Rate for various SD memory cards achieved with *'Read Multiple Block'* command.

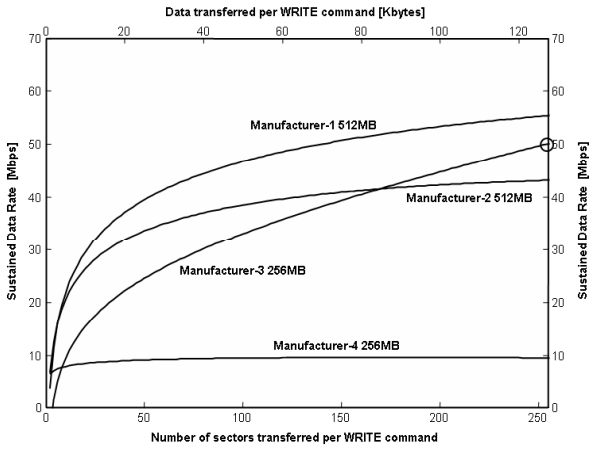


Fig. 7. Sustained Write Data Rate for various SD memory cards achieved with *'Write Multiple Block'* command.

*ple Block'* commands respectively. These figures also present how the card's supply current is related to the card's status and to its internal operations. The supply current increases when the internal flash memory is accessed.

Based on experimental measurements on various SD cards, their sustained data rate has been determined. Figures 6 and 7 present the data rate that can be achieved in various SD cards during read and write data operations. It has to be mentioned that all these cards satisfy the same peripheral interface but they exhibit different performance since they use different flash technology and/or different internal organization.

## V. Conclusions

In this paper we presented a flexible instrument that is able to perform complicated tests on various widely used peripheral devices, in order to measure and analyze their performance. Its versatility and effectiveness are based on the integration of the Matlab simulation environment with a reconfigurable hardware platform, where the 'device under test' is connected. The proposed set-up can be utilized for evaluating, resolving and comparing the performance of existing peripheral devices. Since it provides the means for generating any type of test sequences and scenarios, it can also be used for the development of new peripheral interfaces.

## Acknowledgments

This work was financially supported by the IBM Zurich Research Laboratory.

## References

[1] Richard M. Sherwin, "Memory on the Move", *IEEE Spectrum*, pp. 55-59, May 2001.

[2] The MultiMedia Card, System Specification, Version 3.31, March 2003.

[3] SD Memory Card Specifications, Part 1, PHYSICAL LAYER SPECIFICATION, Version 1.0, March 2000.

[4] ATMEL Corp., AN-4:Using Atmel's DataFlash Application Note, Rev. 0842D, November 2002.

**Panayiotis Savvopoulos** received his Diploma in electrical engineering and computer technology from the University of Patras, Greece in 2001. Since 2001 he is a graduate student at the Department of Electrical Engineering of the University of Patras, Greece. His research interests are in the area of digital communications and specially on iterative satellite receivers. Mr Savvopoulos participates in various R&D projects of European industries.

**Maria Varsamou** is a graduate student at the Department of Electrical Engineering of the University of Patras, Greece. Her research interests are in the area of digital communications,with emphasis on error control coding. Mrs Varsamou participates in various R&D projects of European industries.

**Theodore Antonakopoulos** is an Associate Professor at the Electrical Engineering Department of the University of Patras, Greece. His research interests are in the area of digital communications with emphasis on performance analysis, efficient hardware implementation and rapid prototyping. He has more than 100 publications in the above areas and is actively participating in several R&D projects of European industries.