Reprint

Prototyping and Testing of the Secure Digital Interface

P. Savvopoulos, M. Varsamou and T. Antonakopoulos

5th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2006

PATRAS, GREECE, JULY 2006

Prototyping and Testing of the Secure Digital Interface

Panayiotis Savvopoulos, Maria Varsamou and Theodore Antonakopoulos

University of Patras
Department of Electrical Engineering
26500 Rio - Patras, Greece
{psavvop, mtvars, antonako}@upatras.gr

Abstract

Secure Digital (SD) is one of the most popular interfaces used in storage and other peripheral devices in a wide range of consumer products. This paper presents the prototyping of a SD interface module, analyzes its architecture and describes the used implementation methodology. The implementation methodology is based on a versatile hardware/software co-design platform. In order to support the prototyping of the SD interface, a flexible testing tool was developed and used during the prototype's verification and validation procedures. The testing tool combines the high-performance modelling capabilities of Matlab and a reconfigurable device platform.

1 Introduction

The market trend for portable consumer devices with large storage capabilities along with the need of new consumer applications has led to the growth of the peripheral devices market. Crucial characteristics of these peripherals are their portability and compatibility with a variety of consumer products, due to the widespread use of standardized interfaces [1]. Great effort is devoted to the development of consumer products that satisfy the need for faster, cost-effective and more reliable consumer peripherals. As a consequence, effective interfaces that support high sustained data rates need to be developed. This key feature of peripheral devices determines their overall efficiency and as a result, their market success.

In this paper, we discuss the prototyping and testing of a module of the Secure Digital (SD) interface, that is used in several peripheral devices such as memory cards [2]. A detailed implementation approach is introduced that analyzes and translates the interface functional requirements into modules of a hardware/software co-design prototyping platform. The software modules are executed on an IBM PowerPC (PPC) processor [3], while the various circuits are implemented as hardware peripherals.

For the functional verification of the SD interface module under all possible data exchange conditions, the appropriate testing environment has to be employed. Although commercial host adapters could be used to validate the basic functionalities of the implemented interface, the performance of sophisticated low-level tests for a more thorough investigation is impossible, as they only allow high-level and constrained access to an SD device, through several operating system dependent I/O drivers. Therefore, a complete versatile testing environment was developed that combines the powerful modelling capabilities of the Matlab tools with a reconfigurable hardware platform attached to the PCMCIA interface of the host computer [4]. This environment gives to the designer the capability to define elaborate test sequences and to investigate extensively the performance of any device that uses the Secure Digital interface. The developed setup is a complete prototyping and testing solution for the implementation and verification of the SD interface. As it is based totally on reconfigurable logic, it can be easily modified and, with the minimal of changes, adapted to other consumer peripheral interfaces. Therefore, it constitutes a unified development environment for such type of interfaces.

Section 2 gives an overview of the Secure Digital card interface as it is used in memory cards. In Section 3 the prototyping methodology and the hardware/software co-design approach are discussed. Finally, in Section 4, we highlight the testing environment that was developed for the extensive verification of the prototype's functionalities and the evaluation of its performance.

2 SD Card Interface Fundamentals

The SD interface is commonly used in consumer devices that contain removable storage cards like PDAs, Digital Cameras, MP3 players etc.. The interconnection of a

SD Card with a Host device is based on an advanced 9-pins physical interface (Clock, Command, 4xData and 3xPower lines) and some protocol engines in the form of Finite State Machines (FSMs) [2]. Two types of FSMs exist and control the operations of the SD interface. The first type is responsible for manipulating the detection/initialization procedures, while the second one controls the steady-state and data transfer operations. During the SD initialization procedure, the SD Card and the Host exchange control information, the SD Card is detected and the Host identifies the features supported by the card. When the initialization procedure is completed and the corresponding FSM has reached its last state successfully, the Card enters the normal operation, meaning that data transfers from/to the Host device can be performed.

The interaction between the Host device and the SD Card and the evolution of the Card's FSMs, is achieved using specific data bit-streams, while the Host supervises the data exchange, issues commands to and receives responses from the peripheral device, over the dedicated Command line. The command is used by the Host as a token that triggers the Card interface circuits to perform an operation. After processing the Host command, the Card evolves its FSMs and responds accordingly.

The Host commands are divided into two main categories defined in the SD memory Card Specifications:

- The first category includes commands that are used for the exchange of control information and configuration parameters between the Card and the Host, during initialization and normal device operation. Most commands are related with respective Card responses, except a few that do not require any response from the Card.
- The second category concerns commands that initiate and control data block transfers between the two communicating ends. The data are transferred serially through the Data lines of the physical interface.

3 SD Card Interface Prototyping Methodology

The initial step in the development of the SD interface prototype involves the functional decomposition of various procedures into separate but cooperating hardware and software components. This approach must comply with the SD Card Interface specifications and also take into consideration the prototyping platform capabilities and constraints. The determination of the required hardware and software submodules comprises a very critical task, since it determines the overall performance and efficiency of the final system. After completing the analysis of the procedures and mechanisms that take place during initialization and normal

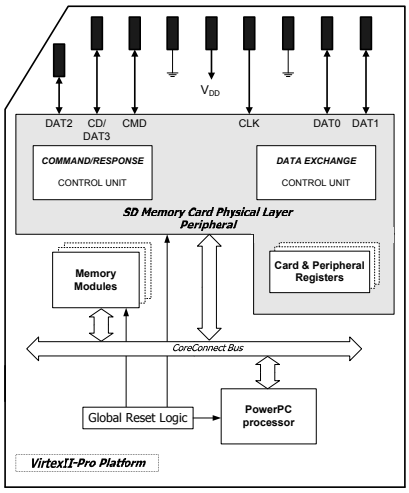

Figure 1. SD Interface Prototype Platform Architecture.

operation of the SD Card interface, differentiation is made depending on the time criticality of each event. For the development of our SD interface, a flexible approach was followed, using FPGA reconfigurable logic with an IBM PowerPC (PPC) core processor. Hardware modules represent tasks with limited execution and response time, while other tasks along with the control of the complete prototype system are implemented in the form of PPC code. The hardware circuits are designed to manipulate and properly translate the signals appearing on the SD interface and at the same time, to support the tasks executed by the PPC processor. A general description of the developed prototype architecture is given in Figure 1.

More specifically, the implementation was based on a Virtex-II Pro Prototype Platform [5], using a Virtex-II Pro device that incorporates an embedded IBM PowerPC 405 RISC processor as a hard IP block [3]. The platform is ideal for designing systems that include IP cores and customized peripheral hardware modules of reconfigurable logic, which are attached to the PowerPC processor through the IBM CoreConnect bus architecture [6] as slaves peripherals. The described platform provides the flexibility to explore several configurations concerning different tasks that can be translated into several software or hardware submodules, as either PPC code functions or dedicated hardware circuits respectively, depending on application specific criteria.

The SD peripheral (Figure 2) keeps PPC informed about the occurring events on the physical layer, i.e. the reception of a new command. This is feasible through the *'Card & Peripheral Registers'* submodule of the Interface peripheral, and the dedicated hardware circuits of the *'Command/Response Control Unit'* submodule which processes

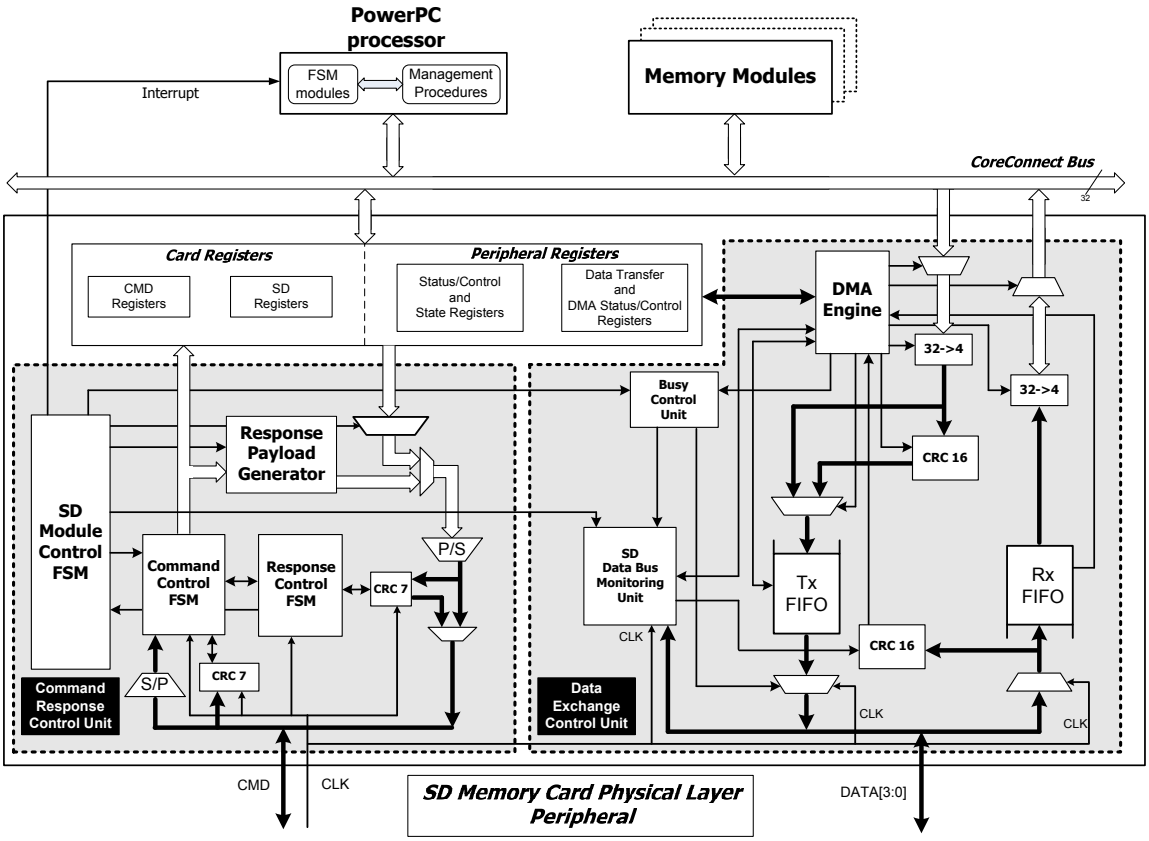

Figure 2. SD Memory Card Interface Peripheral Architecture.

the bit-streams transmitted over the interface. Then, the PPC processor is informed about the pending command and serves the requested task. The PPC is responsible for updating the contents of several registers in the *'Card & Peripheral Registers'* submodule. The *'Command/Response Control Unit'* manages the generation and transmission of the response bit-streams, which include the necessary information collected from various registers of the SD submodules. The *'Command/Response Control Unit'* can be considered as a general-purpose module that with minor changes can serve other similar interfaces, such as MMC, SDIO etc, which are based on the logic of serial exchange of Commands/Responses between a peripheral and a Host device.

The other hardware submodule, which is called *'Data Exchange Control Unit'*, deals with functions performed during normal operation, i.e. exchange of data blocks. A representative example of a data transfer procedure as it is performed inside the SD Memory Card prototype is given in Figure 3. After the exchange of the *'Write Multiple Block'* command and the respective response between the Host and the prototype, the above unit is activated and takes the control of the Interface Data Bus, in order to transfer the data appearing on the interface to the PPC memory modules. This is accomplished with the contribution of a custom DMA engine that transfers parts of the transmitted blocks as they are stored in the Rx FIFO.

Table 1 shows resource allocation and utilization details for the implementation of the SD Card Interface specific peripheral in the Virtex II Pro device.

4 SD Interface Testing Environment

In order to validate and verify the proper functionality of the SD Card physical layer implementation a flexible environment that is capable of performing complicated testing scenarios, has been developed [4]. Although there are commercial host adapters that allow SD Cards to be accessed by a host computing device, it is very difficult, and sometimes impossible, to perform elaborate tests for verification purposes using these adapters, since they only allow high-level access to the device through a stack of operating system dependent I/O device drivers. On the contrary, the developed testing environment provides low-level control and enables the execution of elaborate test scenarios. Systematic and methodical debugging can be performed leading to

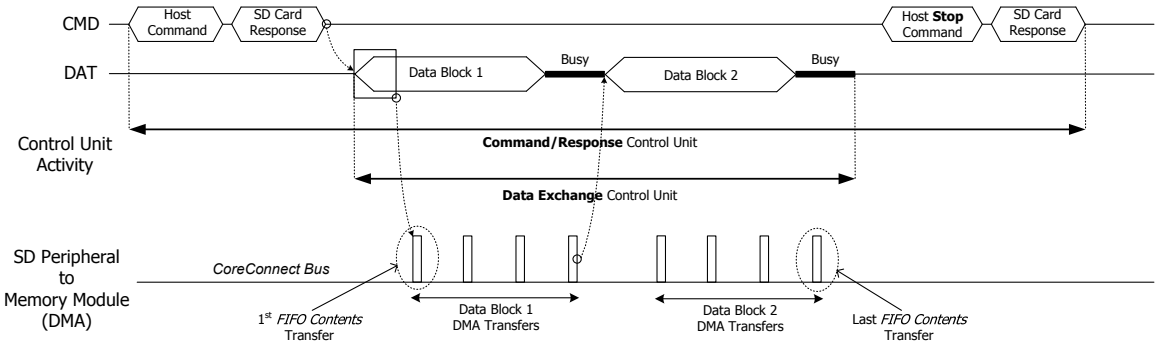

Figure 3. An example of the *'Write Multiple Block'* Command in the SD Memory Card Prototype.

Number of External IOBs	34	13%
Number of LOCed External IOBs	34	100%
Number of PPC405s	1	100%
Number of RAMB16s	21	47%
Number of SLICEs	2909	59%
Number of BUFGMUXs	1	6%
Number of JTAGPPCs	1	100%
Number of TBUFs	8	1%

Table 1. Virtex II-Pro resources utilization.

error identification and correction early in the design process. The proposed environment consists of an integrated set-up that combines the high-performance simulation and modelling capabilities of Matlab tools with a flexible and reconfigurable hardware platform that is attached to a conventional laptop computer via the PCMCIA interface. The basic architecture of the testing environment is presented in Figure 4. The Matlab environment communicates with the hardware platform via a custom PCMCIA interface that provides data exchange and synchronization between the FPGA-based circuits and the Matlab workspace. This interface involves the implementation of a custom I/O device driver and custom logic that extends the PCMCIA's accessibility. The reconfigurable hardware platform, that is based on FPGA circuits of a Virtex II device [7], acts as a host adapter that actually issues the appropriate bit-streams to the device under test, according to the scenarios implemented as Matlab scripts. 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.

The FPGA-based hardware platform implements the adaptation control unit required for interconnecting the peripheral device with the high-level modelling tool. The hardware platform's internal logic is mostly based on the implementation of multiple, functionally-related FIFOs for storing and transmitting data from the Matlab workspace to the peripheral device and vice versa as well as for recording timing information about specific events for debugging and validation purposes. 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.

4.1 Testing Scenarios Implementation

Various basic functions of the SD memory Card 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 testing scenarios that generate a sequence of application specific commands, but still remaining independent of the characteristics of the specific physical interface. The Matlab environment has the control of the entire system and of the status of the peripheral device throughout the execution of a testing sequence script.

For the execution of a specific testing scenario the ap-

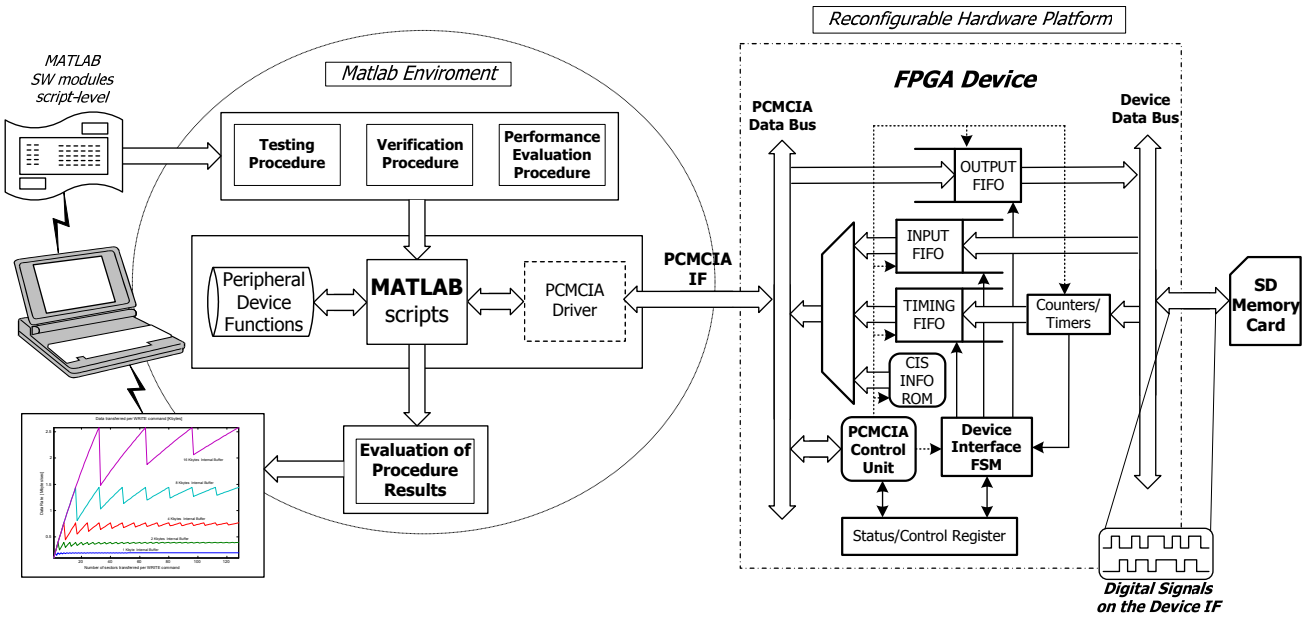

Figure 4. SD Card Interface Testing Environment.

propriate scripts are initially implemented in the Matlab environment. During their execution they exchange essential control and data information with the hardware platform, which in turn translates the received data into signal waveforms compatible with the SD Memory Card physical layer 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. Additional internal logic acquires timing information between the various interface events. This information is finally retrieved in the Matlab workspace and can be further evaluated. As a consequence, the device's proper functionality can be verified under both normal and erroneous circumstances and useful conclusions about the performance of the developed device can be drawn.

5 Conclusions

In this paper, we introduced a flexible setup for prototyping and testing the Secure Digital (SD) Interface used by various consumer products and peripherals. The proposed setup consists of a prototype platform and a supplementary testing environment used for validating the developed system. The prototype platform is based on reconfigurable logic, where the several operations are carefully mapped into cooperating hardware/software modules. Additionally, a complete testing environment that emulates a Host device and enables the execution of various testing scenarios is also presented. The modularity and flexibility of the proposed setup provide the means for effective, rapid prototyping and testing of a wide variety of consumer peripheral interfaces.

6 Acknowledgments

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

References

[1] R. M. Sherwin, "Memory on the move," *IEEE Spectrum*, pp. 55–59, May 2001.
[2] SD Memory Card Specifications, Part 1, PHYSICAL LAYER SPECIFICATION, Version 1.0, March 2000.
[3] PowerPC 405 Embedded Processor Core User's Manual, Fifth Edition, December 2001.
[4] P. Savvopoulos, M. Varsamou and T. Antonakopoulos, "A Versatile Instrument for Analyzing and Testing the Interfaces of Peripheral Devices," *The 3rd IEEE International Conference on Systems, Signals & Devices - SSD'05*, Sousse, Tunisia March 2005.
[5] Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet, Version 4.3, June 2005.
[6] The CoreConnect Bus Architecture, September 1999.
[7] Virtex-II Platform FPGAs: Complete Data Sheet, Version 3.4, March 2005.