Available online at www.sciencedirect.com
SCIENCE DIRECT
Microprocessors and Microsystems 29 (2005) 421–434
MICROPROCESSORS AND MICROSYSTEMS
www.elsevier.com/locate/micpro

Multiprocessor SoPC-Core for FAT volume computation

Armando Astarloa*, Unai Bidarte, Jesús Lázaro, Aitzol Zuloaga, Jagoba Arias

Department of Electronics and Telecommunications, Faculty of Engineering, University of the Basque Country, Urquijo s/n, E-48013 Bilbao, Spain

Received 20 October 2003; revised 4 May 2004; accepted 5 January 2005
Available online 7 February 2005

Abstract

This paper presents the design, co-simulation and implementation of a Soft-core for the autonomous reading of a file stored into IDE devices formatted with FAT16 File Data System. This application illustrates a novel core architecture that embeds multiple customized *tiny* microprocessors and standard interfaces into the core. The reconfigurable nature of the FPGA implementation allows easy modifications of the microprocessors and peripheral hardware to cover other control applications. Emphasis is placed on presenting how the co-design and co-simulation of the processors, additional hardware, buses and communications has been possible with the developed specific Virtual Environment.

Keywords: IP Core; SoPC; FPGA; VHDL; Co-design

1. Introduction

Nowadays technology is able to manage millions of logic gates in a single integrated circuit working at frequencies of up to 1 GHz. System-on-Chip (*SoC*) design methodology is evolving very fast in order to make these chips a reality. Rajsuman defines the SoC concept as "an integrated circuit designed connecting multiple independent VLSI blocks, that performs all the functionalities of the application" [1]. This definition emphasizes the importance of the previously designed and verified blocks. Borel gives the following definition for the same concept: "an integrated circuit that integrates process, control, storage, communication, sensor and actuation capacities" [2].

A generic *SoC* could include a microprocessor or microcontroller, memories (SDRAM, DRAM, ROM, FLASH, CAM, etc.), clock synchronization resources (DLL or PLL and clock distribution buffers), communication interfaces (USB, PCI, IDE, RS232, EPP, CAN, etc.), data converters (A/D or D/A), input/output units, direct memory access controllers and application specific cores.

VLSI market imposes short design cycles, high complexity and high performance. Development time can be enlarged only as far as time-to-market parameter allows, which usually is not what designers would desire. And although having a short designing windows, complexity and performance cannot be reduced, in order to maintain the competitiveness. These design requirements are becoming more and more crucial, and complex *SoC* design methodology is evolving very fast, for the designers to have a powerful design method to manage the challenges of the technology.

There are two main strategies to face the technology gap. The groundwork of the first one is the elevation of the hardware description abstraction level, using system level languages. The second one is based on reusing previously designed and verified Intellectual Property (*IP*) blocks, known as IP Cores. The first one requires advanced and high-level hardware description and synthesis tools, whereas the second one requires extra time in the cores description process in order to get easily reusable blocks.

The design team in which the authors of the article work has several years experience in applying the second strategy. An extensive library has been created including predesigned and preverified cores with a specific functionality but useable in many application fields. The core described in the present work is also present in this library.

* Corresponding author. Tel.: +34 94 601 7304; fax: +34 94 601 4259.
E-mail address: jtpascua@bi.ehu.es (A. Astarloa).

0141-9331/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2005.01.001

The cores, also named Virtual Components, may come in several forms. They may be hard, with all the gates and interconnections placed and routed and all the silicon layers defined. Hard-component performance is thus predictable. Other Virtual Components can be soft, with only the higher level, register transfer level representation defined. Such components must be synthesized, placed, and routed in the physical representation. Soft Virtual Components are easy to modify and facilitate reusability, but require more verification, take longer to implement, and have less predictable performance than hard Virtual Components. Finally, a third variety of Virtual Components, firm, comes as a register transfer level description with some level of physical floorplanning or placement, but not the final routing. These Virtual Components provide more predictable performance than soft components, but still allow some flexibility in block shape and final placement [3]. For all cores, a commonly accepted assumption in core-based system design is that all new designs start with a description in one of the two popular hardware description languages: VHDL or Verilog. Our goal is to design *SoPC* cores that maximize the future reuse possibilities, so the core described in this work, as well as most of the cores included in the previously noted library are soft cores. From now on, whenever the term core is used, it will make reference to a soft-core.

Previously, IP Cores used non-standard interconnection schemes that made them difficult to integrate. This required the creation of custom glue logic to connect each of the cores together. Some years ago the design team decided to adopt a standard interconnection scheme to integrate the cores more quickly and easily in the final applications. General purpose interfaces defining the standard data exchange between IP Cores were studied, and the *Wishbone SoC* interconnection architecture for portable IP Cores was selected [4]. These are some reasons that justify the adopted decision: robust standard, facilitates structured design methodologies, independent of the technology, flexible, well documented, master/slave synchronous protocol, completely free and the existence of many users that share their projects using internet.

This work is focused on *SoC* applications using Field Programmable Gate Array (*FPGA*) technology, which is known as System-on-Programmable-Chip (*SoPC*). In this way, only the digital part of the system can be integrated in this chip. This digital part is what usually can be integrated in the so called *SoC* device even when using ASIC technology. The reconfigurable nature of the FPGAs allows fast and simple modifications of the proposed architecture to cover other control dominated applications.

2. Motivation and related work

There are many systems that need a portable massive storage device as source of data. Well known examples are digital cameras, MP3 players, electronic data acquisition systems and embedded systems in general. Furthermore, it is usually necessary, sooner or later, some processing of the data stored in the storage device, in most cases using a Personal Computer. That constraint makes necessary the use of the File System adopted by the operating system that runs on the PC to organize the data into the storage device. Embedded systems based on powerful microcontrollers can manage the communication with the storage device and the File System processing the same way they manage any other task. Depending on the application, the overhead produced by this task can be acceptable, especially if the later processing of the read data is done at the same CPU. Related work about academic FPGA implemented IDE cores can be found in [5] (ATA/ATAPI-5 modes) and in [6] (PIO mode). Both hardware modules need a host to set the ATA commands to the IDE device and to process the FAT volume. These cores and the commercial ones available in [7] or [8], fit with SoC powerful embedded microprocessors.

But there are many cases where the employment of an independent and specific module for the communication with the storage device and the data extraction from the File Data System might be suitable: embedded designs with high-performance requirements, applications with extensive data exchange requirements, SoC based autonomous DVD rewriters [9], printers with Compact-Flash device support, etc. In all these situations, the use of an independent core which puts the read and processed data into the internal SoPC high-speed bus fits with the system operativity better. On the other hand, the reutilization of the architecture is simplified because if the data source element changes (a different communication channel or another local storage device) it can be easily replaced and the rest of the architecture may be maintained without any modification [10].

A conventional IDE core does not provide the extracted data from a file stored in an IDE volume to an internal on-chip bus. This could be desirable for the mentioned applications. If this operativity want to be resolved by a single core, it must be noticed that the control of the ATA protocol and the latter FAT process are complex tasks. This makes the design of the core using conventional hardware architectures not viable. The required FPGA consumed will be prohibitive with that approach.

The result of the work presented in the article is a novel intra-core architecture that embeds fast and small micro-controllers in conjunction with custom HDL hardware and standard interfaces either for internal and external interconnections. The application of the *reusing* concept to the SoC design methodology forces not only the use of a standard interconnection architecture for IP Cores, but the enhancement of the core portability when the design of the core is performed. To illustrate this architecture an independent core for FAT16 processing is presented.

The design methodology adopted tries to find a good trade-off between the area requested, the speed performance and the reusability facilities.