DIGITAL DATA PROCESSING PERIPHERAL DESIGN FOR AN EMBEDDED APPLICATION BASED ON THE MICROBLAZE SOFT CORE

E.Ostúa, J.Viejo, M.J.Bellido, A.Millán, J.Juan & A.Muñoz

Grupo de Investigación y Desarrollo Digital (ID^2)
Departamento de Tecnología Electrónica - Universidad de Sevilla
Av. Reina Mercedes, s/n (E. T. S. Ingeniería Informática) - 41012 Sevilla (Spain)
Tel.: +34 954556160 - Fax: +34 954552764

email: { ostua; julian; bellido; amillan; guerre; jjchico; paulino } @ dte.us.es

ABSTRACT

In this paper we present a design of a peripheral for MicroBlaze soft core processor as part of a R+D project carried out in cooperation with three different companies. The objective of the project consisted in the development of an embedded system with a SoC implemented on a FPGA custom-designed board. This work addresses the design of a Digital Data Processing peripheral included as a part of the target SoC application, that process digital signals via the digital inputs on a proposed board. Peripheral functionality is configurable for each digital signal independently and is configured from the software running on the MicroBlaze processor core.

1. INTRODUCTION

The continuous development of the integrated circuit technology is leading to an increasing integration density besides an improvement of the operation speed of electronic systems. In digital systems, it allows for huge performance improvements while the system size shrinks more and more. However, design methodologies are also increasing in complexity, and new challenges are to be faced by designers. Two fundamental problems appear: on one hand, the increment in speed (switching activity) and device density translates in greater power consumption and power and thermal density that may induce system failure unless effective cooling mechanisms are included; on the other hand, as integration density increases, the total number of devices included in the chip increases (up to several million devices these days). Then, it becomes necessary to define good design and verification methodologies in all the levels of abstraction in order to handle effectively the complexity of the whole problem.

One of the more extended methodologies in this sense is the so-called System-on-Chip methodology or, more simply, SoC design. The basic idea under this methodology is to use the current integration possibilities not to design more complex specific systems, like Intel or AMD processors, but to implement whole systems made out of several individual building blocks on a single chip. This way, systems currently implemented out of several chips on a PCB (Printed Circuit Board) are being integrated in a single chip while maintaining the overall structure of the system. This methodology provides many advantages without the need to completely re-define the way electronic designers work. Among these advantages are power consumption reduction, increased performance and much more miniaturization, making the SoC design a strong methodology for embedded and ubiquitous applications.

Almost 100% of the SoC's are digital systems built around a microprocessors that acts as a controlling unit. The selected microprocessor will strongly condition the performance and overall characteristics of the system, and the cost of the SoC. It is important to note that most embedded applications do not require high computation capabilities, but usually demand high reliability and low power consumption.

There are a several processor cores that are commonly used in SoC applications, both privative cores that requires the acquisition of a license of use, like ARM [1], PowerPC [2], NIOS3 [3], MicroBlaze [4], and many others; and free or open cores that may be used without the need to acquire a license, like LEON3 [5] and OpenRisc [6]. The main advantage of privative processors are that they are usually well tested and optimized for a specific target hardware and provide a complete set of CAD tools to make the SoC design an easier process. For example, MicroBlaze from Xilinx is well integrated with the development platform from the same foundry, which leads to highly optimized designs at the cost of being bound to a particular technology (Xilinx Spartan and Virtex FPGA families [7]) and a concrete set of tools (Xilinx ISE and EDK [8]).

This paper is the consequence of latest developments of the author's research team in the SoC design area. A detailed view of the design and implementation of the digital data processing function for industrial applications is presented.

978-1-4244-1992-0/08/$25.00 ©2008 IEEE.

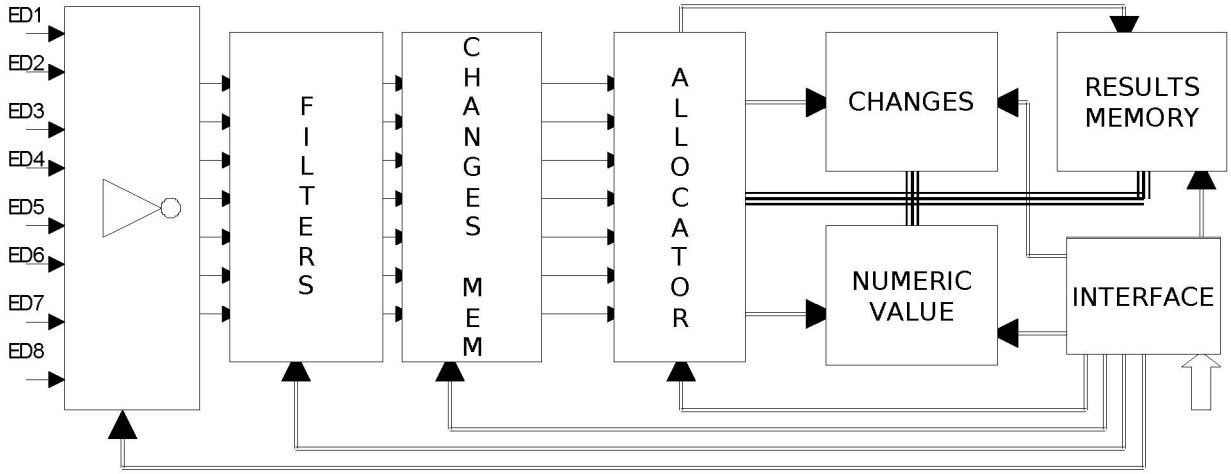

Fig.1. Digital Data Processing Peripheral functional diagram

The peripheral is part of the OpenRTU project, a company effort to implement a System-on-Chip application leaded by a MicroBlaze processor core that will run on a Spartan-3 FPGA from Xilinx, on top of a custom-designed board. The main functionality of the SoC system is the digital signal (DSP) and digital data processing. That is accomplished with two different peripherals implemented as IP cores on SoC and so embedded on the same FPGA as the MicroBlaze. The *digital data processing peripheral* is the object of this study.

The rest of the paper is organized as follows: in the next section we provide a brief introduction to the peripheral by discussing the specifications. Section 3 describes the design and implementation methodology including the software and hardware design process and application integration. Section 4 presents the main results obtained in this project. Finally, some conclusions are derived in section 5.

2. PERIPHERAL SPECIFICATIONS

The objective peripheral specifications are to process digital signals, acquired directly via digital inputs on the board, through a group of different functions. On Fig.1 a diagram of the peripheral main components is shown, with a module for each processing function.

Peripheral functionality is configurable for each digital signal independently in several parameters, which includes disabling a signal passing through some of the processing functions, timing configuration on filters and memories, input signals grouping configuration, status registers on the inputs and others.

Operation on the digital peripheral is configured from the software running on the MicroBlaze processor core. This goal is committed in the Peripheral Control box, where a series of global configuration registers are available to be read and written with a pre-designed interface to the SoC OPB bus, where MicroBlaze communicates with others IPs in the design including this peripheral. There is also some configurable interrupt handling done in this core, in order to let MicroBlaze know when a concrete process is completed. Also a polling of the information is possible from any OPB peripheral. The results of each instant process are done available to the MicroBlaze processor who then captures the data, archives it in the RAM memory and finally the software running process a large number of information.

As described above, each of the digital inputs can pass through several function modules in a chain, where a different task is committed on each signal independently or on grouped signals in each case.

The first module in the chain is a *digital inverter* where each signal is configured to be inverted or not, depending on the digital input work conditions to the DSP board.

Then the signal is passed through the *filter*, where the digital signal pulses with a width less than configurable parameter in milliseconds will be ignored.

Next module is the *changes memory*, so after every change in the signal, it will remain unaltered for a configurable period of time in milliseconds.

The last signal processing module is the *allocator*, with a quite complex functionality. It first reorders the signals according to the programmable configuration in order to describe each signals real life functionality, so each digital input can be a single signal, a double signal (grouped with another input), or a part of a group of signals in order to collect misc digital data, like counters or numeric values. Next, each group of signals obtained is treated to identify the nature of the values it concerns, again a configured behavior, activating some error flags if necessary and finally the results are stored on the corresponding output registers. These tasks are accomplished together with the *counters*, *numeric value* and *results memory* modules from the peripheral.

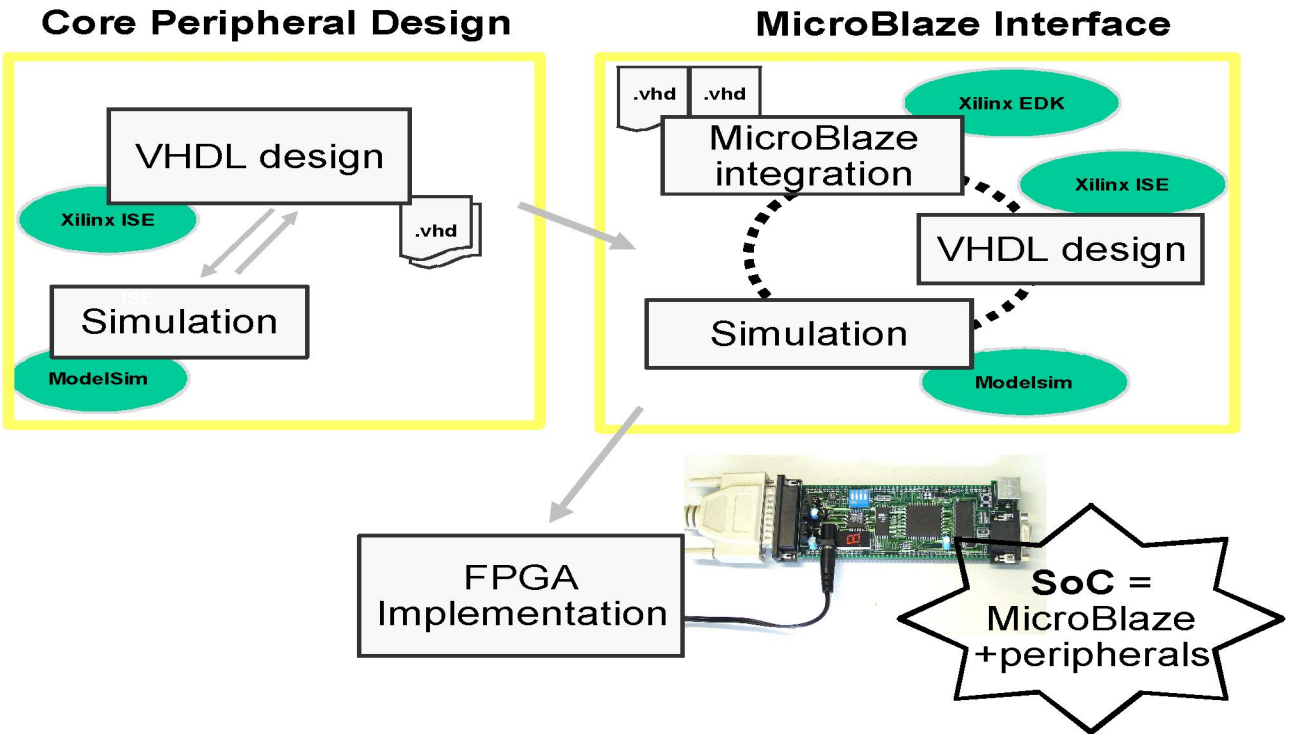

Fig. 2. SoC design methodology

So, in the *counters* module, each pulse on the input is used to increment a predefined counter associated to that digital signal.

The *numeric value* module is used to convert a configurable group of inputs to a numeric value as the resulting 8-bit word, in one of the following output formats: 2 BCD values, 1-of-8 word or regular binary.

The *results memory* module goal is to store the main results of the processing modules so they can be available in time when MicroBlaze requests them.

3. PERIPHERAL DESIGN AND IMPLEMENTATION

In this section the more important concerns in the design and implementation of the peripheral are described in detail.

After the analysis of the peripheral and complete project specifications, it was taken into consideration the design methodology to be used and so what the flow of the design process will be. As the complexity of the design is not too high and the functionality is pretty well defined and divided in several modules, the decision was to design the core of the peripheral in synthesizable VHDL, using the Xilinx ISE Foundation IDE tool in order to synthesize the model and implement it to the target FPGA family. The simulator used was Modelsim, by Mentor. Also in order to design the interface of the peripheral to MicroBlaze via the OPB Bus the tool Embedded Development Kit (EDK) by Xilinx was used. Fig.2 illustrates the SoC design methodology used in this project.

Each functional module was written in VHDL, simulated and synthesized individually and then a latest module was designed to glue all the logic, including configuration and output registers and also interconnect the peripheral as a slave of the OPB bus so it can communicate with MicroBlaze.

In order to model each functional module of the peripheral, each one was designed as a canonical finite state machine written completely in VHDL. Once each block ends its processing it activates a signal that initiates the next one in the chain. After the last step has finished main results are archived and are available to MicroBlaze, which can read them anytime before the next set of digital inputs are processed. Also it's possible to send an interrupt message to the SoC interrupt controller when new results are available.

All the elements in the peripheral work with a single clock signal, at a maximum frequency of 90 Mhz, as shown on the results chapter, a value over the specifications. As the timing specifications were not so restrictive and area consumption was important, also big effort was spent on area optimization by reutilization of the complex functions for each signal and doing a serial processing when possible.

The peripheral initialization is done in hardware with a pulse of a single reset line and it can also be done in software by the program running on the MicroBlaze core.

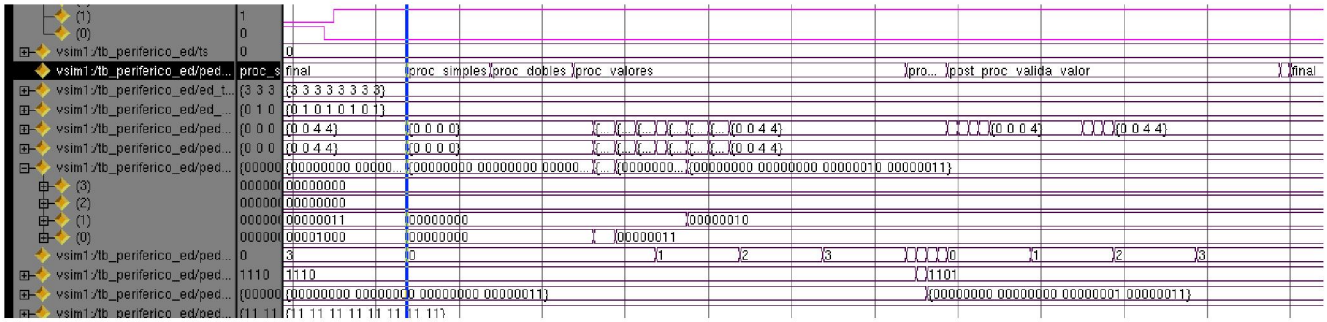

Fig. 3. Simulation of the peripheral

4. RESULTS

Main simulation and implementation results are discussed in the chapter.

Simulation was completed for each functional module independently in the peripheral and later on the complete core. The interface to MicroBlaze was then added to the design and evaluated and synthesized with Xilinx EDK tool. It was finally included in the completed design for an on-chip verification, including others peripherals and IP cores developed for this SoC application. A screen capture of the simulation process on the complete peripheral is shown in Fig. 3.

The synthesis and implementation of the peripheral was done with Xilinx ISE and EDK toolkits and main results of this process are shown in Table 1 for the target architecture.

Table 1. Implementation Results for a Spartan-3 1500

	slices	total	%
peripheral core	397	13312	2
peripheral & microblaze	984	13312	7
max operational freq.	92 Mhz		

5. CONCLUSIONS

In the paper we have presented the design and implementation of a digital data processing peripheral for Microblaze as part of the SoC of a embedded system. We used a methodology based on direct VHDL codification, including an interface to the microprocess core provided in part by the partner tools, which results in a very good performance for the designed peripheral. Effectively, results shows that the design occupies less than 2% over a 1500 Spartan-3 and the speed operation is higher than 90Mhz.

6. ACKNOWLEDGMENTS

This work has been partially supported by the Ministry of Education and Culture of the Spanish Government through the TEC2007-61802/MIC (HIPER) project and the Andalusian Regional Government's EXC-2005-TIC-1023 project.

7. REFERENCES

[1] Steve Furber: "ARM system-on-chip architecture, 2nd edition", Ed. Addison-Wesley 2000.

[2] "IBM PowerPC Quick Reference Guide", IBM Corp. 2005.

[3] "NIOS 3.0 CPU Data Sheet", Altera Corporation, 2004, http://www.altera.com/literature/ds/ds_nios_cpu.pdf

[4] "Microblaze Processor Reference Guide", Xilinx Inc. 2005, http://www.xilinx.com/ise/embedded/mb_ref_guide.pdf

[5] Jiri Gaisler, Sandi Habinc, Edvin Catovic: "GRLIB IP Library User's Manual", Gaisler Research, 2006, http://www.gaisler.com/products/grlib/grlib.pdf

[6] Damjan Lampret: "OpenRISC 1200 IP Core Specification", 2001,http://www.opencores.org/cvsget.cgi/or1k/or1200/doc/or1200_spec.pdf

[7] "Xilinx FPGA Silicon Devices", Xilinx Inc. 2006, http://xilinx.com/products/silicon_solutions/fpgas/

[8] "Xilinx Logic Design and Embedded Design Tools", 2006, http://xilinx.com/products/design_resources/design_tool