International Conference on Control, Engineering & Information Technology (CEIT'14)
Proceedings - Copyright IPCO-2014, pp.146-151
ISSN 2356-5608

Impact of MicroBlaze FPGAs Design Methodologies of the Embedded Systems Performances

I. MHADHBI[1], S. BEN OTHMAN[2] and S. BEN SAOUD[3]
LSA Laboratory, INSAT-EPT, University of Carthage, TUNISIA
imene.mhadhbi@gmail.com
slim.BenOthman@ept.rnu.tn
slim.bensaoud@gmail.com

***Abstract*— Field Programmable Gate Array (FPGAs) present one of the attractive choice for implementing embedded systems due to their intrinsic parallelism, their fast processing speed, their rising integration scale and their lower cost solution. The growing configurable logic capacities of FPGA have enabled designers to handle processors onto FPGA products. In contrast to traditional hard core, soft cores present one of the attractive processors for implementing embedded applications. They give designers the ability to adapt many configured elements to their specific application including memory subsystems, interrupt handling, ISA features, etc. Performance evaluation of these cores presents the great dial of embedded designers to face up the various problems related to the selection of the efficient soft core FPGA embedded processor configuration, against a specific design methodology. The purpose of this paper was to evaluate the impact of the features of Xilinx MicroBlaze processor on the execution time of a lightweight cryptographic application.**

Keywords—***Embedded Systems, FPGAs, MicroBlaze processor, Performance evaluation, Lightweight cryptographic application.***

I. Introduction

Embedded systems are now present in practically all domestic and industrial systems (appliances and applications) such as cellular telephones, personal digit assistants (PDAs), digital cameras, Global Positioning System (GPS) receivers, defence systems and security applications. The increased complexities of embedded systems, and their real-time operations constraints, allow semiconductor markets to build other solutions for processing. Traditionally, embedded systems were designed and realized using microprocessors (MP), microcontrollers (MCUs), digital signal processors (DSPs), application specific integrated circuits (ASICs) and FPGAs. Continuing increases in FPGA performance, capacity and architectural features are enabling more embedded systems designs to be implemented using FPGAs. Additionally, FPGAs costs are decreasing, allowing designers to incorporate FPGAs circuits with one million equivalent gates for less than $12 [1].

The emergences of the embedded processors on FPGAs have increased their utility. It presents one of the most exciting developments in FPGA. It is possible, now, to create a complete hardware and software system with I/O and control interfaces on a single chip. Embedding a processor inside an FPGA has many advantages: Hard core processor has the advantage of high computing performance, smaller size. In contrast, soft core processor offers less computing performance with a greatly enhanced in term of configuration, portability, and scalability. The embedded soft core processors defined as software implemented on a hardware in order to realize real-time applications. Several soft core processors are available: Nios from Altera, MicroBlaze and PicoBlaze from Xilinx, LEON2 from Gailer Research and OpenRISC from OpenCores. They are having attracted an always increasing interest due their different advantages:

- Hardware Acceleration: The most compelling reason for FPGA embedded processor is their capacity to make trade-offs between software and hardware to maximize the performance of the embedded system. Implementing the application that is identified as bottleneck, as a co-processor, allows designers to accelerate the execution time.
- Peripheral Personalization: FPGA embedded soft core processor allows complete flexibility for the selection of any combination of peripherals, features or controllers.
- Component and cost reduction: With the high integration scale of FPGA, over 100 soft core soft core processor can be implemented onto single FPGAs. Reducing the components in the design can save design time-to-market and cost.
- Component Obsolescence improvement: Obsolescence improvement is a difficult issue when a design constraint must ensure that a product lifetime is much longer than the typical lifetime of a standard electronics product: Soft FPGA embedded processors could be an excellent solution since the HDL source code for the soft processor can be purchased by guaranteeing the lifetime of the product.

The great dial of embedded systems designer are faced up with the various problems in selecting the best configuration of the soft core processor to implement complex applications in term of energy consumption, hardware resources utilization and the execution time.

International Conference on Control, Engineering & Information Technology (CEIT'14)
Proceedings - Copyright IPCO-2014, pp.146-151
ISSN 2356-5608

The goal of this paper is to evaluate the performance of the different configurations of the Xilinx MicroBlaze soft core processor. The remaining parts of this paper are organized as follows: Section 2 presents the MicroBlaze soft core processor. Section 3 illustrates our design methodology. Section 4 presents the lightweight cryptographic benchmark. Section 5 determines performance results of the Xilinx MicroBlaze architectures. Finally, Section 5 summarizes the paper and gives our perspectives.

II. Overview of the MicroBlaze embedded soft core Processor

The major advantage of choosing MicroBlaze for our research is that we can immediately benefit from the high configurability of this kind of FPGA processor. MicroBlaze soft core processor is a 32-bit Reduced Instruction Set Computer (RISC) architecture optimized for synthesis and implementation into Xilinx FPGAs with a separate 32-bit instruction and data buses to execute programs and access data from both on-chip and external memory at the same time [2]. This processor includes 32-bit general purpose registers, virtual memory management, cache software support, and Fast Simplex Link (FSL) interfaces. It has Harvard memory architecture and uses: Two Local Memory Busses (LMB) for instruction and data memory, two Block RAMs (BRAM) and two peripherals connected via On-chip Peripheral Bas (OPB). Three memory interfaces are supported: Local Memory Bus (LMB), the IBM Processor Local Bus (PLB), and Xilinx Cache Link (XCL): The LMB offers single-cycle access to on-chip dual-port block RAM. The PLB interfaces offer a connection to both on-chip and off-chip peripherals and memory. The CacheLink interface is proposed for use with specialized external memory controllers. The architecture of the Xilinx MicroBlaze FPGA processor, the interfaces, buses, memory and peripherals are shown in Fig. 1.

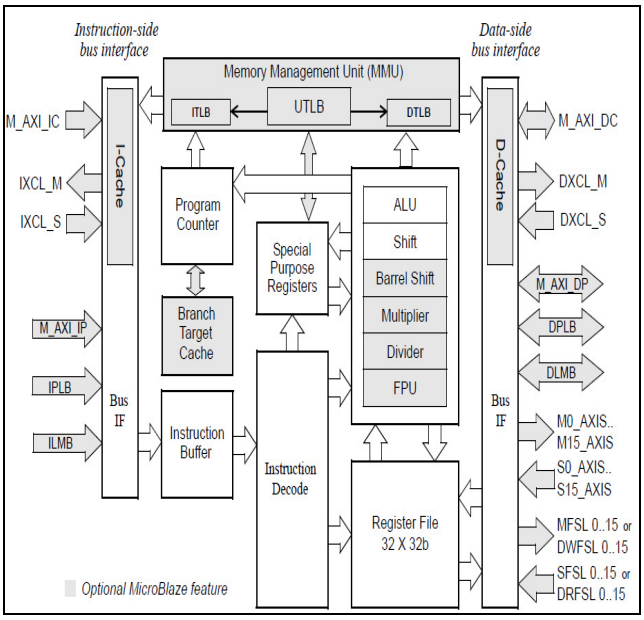

Fig. 1. MicroBlaze Functional Block Diagram [2]

The MicroBlaze Xilinx processor offers designer tremendous flexibility during the design process, allowing them to configure this soft core processor to meet the needs of their embedded systems with adding a specific or fixed set of features required by the design for embedded processor system development. To improve the performance of the MicroBlaze, designer can modify a number of features without change design. They may configure the setting parameters such as:

- Integer Multiplier Units (Mul): Add the Integer multiplication as co-processor.
- Barrel Shifter Units (BS): Add the Shift by bit operations as co-processor.
- Integer Divider Units (ID): Add the Division of Integer numbers as co-processor.
- Floating Point Units (FPU): Add Basic and Extended precision as co-processor.
- Machine Status Register Units (MSRU): Add Set and clear machine status register as co-processor.
- Pattern Compare Unit: Add the String and pattern matching as co-processor.

In the next section, we will introduce the design methodology used to evaluate the different configurations (architectures) of MicroBlaze Xilinx FPGA soft core processor for a lightweight cryptographic application.

III. Used Design Methodologies for the Evaluation of the MicroBlaze Configurations

Embedded system can be implemented using different design methodologies: 1) Full software implementation to given designers the ability to adapt many configurations using the EDK design tool. 2) Full hardware implementation using ISE tool. 3) Both software and hardware methodologies using co-processors to accelerate the design process. 4) High Level Synthesis (HLS) methodology.

Even the rising complexity and time-to-market pressures in the design of embedded systems, designers have moved to the HLS methodology in order to increase productivity. Development of C-Based Design technology has minimized the gap between software developer's experience-level and the knowledge needed to produce hardware implemented applications. Different studies discuss the impact of the design methodology on the performance of the embedded system [3, 4, 5]. They prove that using the HLS methodology can maximize the performance of the embedded system and minimize the time of the design process. It automatically generates a HDL code from a High Level Specification. This specification can be a model (Model-Based Design), a C source code (C-Based Design) or a UML diagram (Architecture Based Design). The most used methodology is the C-Based Design since open sources applications are written on C language.

International Conference on Control, Engineering & Information Technology (CEIT'14)
Proceedings - Copyright IPCO-2014, pp.146-151
ISSN 2356-5608

A. C-Based Design Methodology

C-Based Design methodology allow automatic generation of a synthesized hardware code, VHDL or Verilog, ready to be implemented directly on FPGA or a co-processor, from a C/C++ language [4, 5, 6]. Due to its faster execution speed, the typical design flow for implementing FPGA applications starts with writing them at a high level specification using C language. Fig. 2 presents a comparison of the traditional and the C-Based Design methodologies.

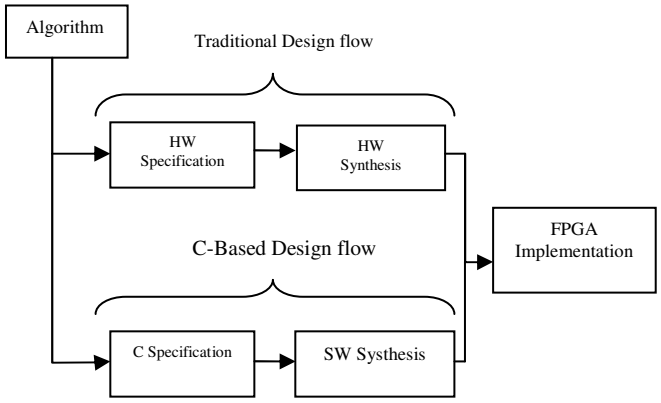

Fig. 2. C-Based Design Methodology

In traditional flows, defining the architecture and generation the HDL code from the C language specification is done manually. This process may require several months to be completed. In the C-Based Design approach, the architecture definition and the generation of the HDL code is often accomplished in a matter of days to week. Many commercial and academic C-Based design tools are found in the literature: Catapult-C (Mentor Graphics) (MG, 2013), CoDeveloper™ (Impulse, 2013), C2H (Altera, 2013), SPARK (Microelectronic, 2013). In our paper, The C-synthesis product used is CoDeveloper™. In the next section, we will present the CoDeveloper™ tool.

B. CoDeveloper™ C-Based Design tool

CoDeveloper™ is a commercialized by Impulse Accelerated Technologies in the CAD market [7]. It allows designers to compile C applications directly into optimized logic ready for use with Xilinx FPGAs, in just minutes or hours. Impulse C code, the input language of CoDeveloper™, can be written and debugged in any ANSI standard C environment, using both fixed and floating point data point types. Impulse C is a library of functions and related data types that give a programming environment, and a programming model, for parallel applications targeting FPGA-based platforms. It has been optimized for mixed software/hardware targets, with the goal of abstracting details of inter-process communication and allowing relatively platform-independent application design. CoDeveloper includes the Impulse C libraries and associated software tools that help you use standard C language for the design of highly parallel applications targeting FPGAs.

IV. Performance Evaluation Process

The performance evaluation of embedded soft core processors has multiple aspects depending on the application that the system is made to. Hence, performance measurement is involved in several stages of the design process, where going back in the process is very costly. In our paper, we will evaluate the performance of the MicroBlaze FPGA soft core processor into a lightweight cryptographic application. In this section, we present, in a first step, performance evaluation techniques, then, we define the application used.

A. Performance Evaluation Techniques

Performance evaluation can be classified into two categories [8]: Performance modelling and performance measurements as mentioned on the table 1.

TABLE I. PERFORMANCE EVALUATION TECHNIQUES

Performance Measurement	MP-on Chip Performance Monitoring counters	
	Off-Chip Hw monitoring	
	Sw Monitoring	
	Micro-coded Instrumentation	
Performance Modelling	**Simulation**	Trance Driven simulation
		Execution Driven Simulation
		Complete System Simulation
		Even Driven Simulation
		Software profiling
	Analytical Model	Probabilistic Models
		Queuing Models
		Marcov Models
		Petri Net Models

Performance modelling approach is concerned architecture-under-development. It can be used at an early stage of the design process where the processor is not available or it is very expensive to prototype all possible processor architectures choices. Performance modelling can be classified into analytical-Based and simulation-Based. Results of analytical modelling approach are not often simple to construct. It allows predict principally user performance, time execution of tasks rapidly without compilation or execution steps. There are not been much studies on analytic approach because their structure is more than those analytical models can be provided. Simulation method presents the most excellent performance modelling method in the performance evaluation of processor architectures. Simulators gives performance information of cycles, execution time, cache bit ratios, branch prediction rates, etc. Results of simulation approach are not very interested to the performance evaluation of the MicroBlaze Xilinx soft core processor because they are not exacted. Designers of this kind of processor have to use the measurement approach to get a fixed measurement of the processors performance which is attempts to implement and verify the architectural and the timing behaviours under a set of benchmark programs.

International Conference on Control, Engineering & Information Technology (CEIT'14)
Proceedings - Copyright IPCO-2014, pp.146-151
ISSN 2356-5608

Several open source and commercial benchmarks are presented. Some of them are: Mibench, Paranoia, LINPACK, SPEC (Standard Performance Evaluation Corporation), and EEMBC (Embedded Microprocessor Benchmark Consortium). These Benchmarks are divided into three categories depending on the application [9]: Synthetic Benchmark (with the intention to measure one or more features of systems, processors, or compilers), Application Based Benchmarks or "real world" benchmarks (developed to compare different processors architectures in the same fields of applications) and Algorithm Based Benchmarks (developed to compare systems architectures in special (synthetic) fields of application).

B. Lightweight cryptographic benchmark: Quark Hash Algorithm

The need for Lightweight cryptographic applications has been frequently expressed by embedded systems designers, to implement a secured applications such as the authentication, the password storage mechanisms, the Digital Signal Standard (DSS), the Transport Layer Security (TLS), the Internet Protocol Security (IPSec), the Random number generation algorithms, etc. Several Lightweight cryptographic algorithms are presented. Lightweight cryptographic algorithms have been designed to fit with a very compact hardware. Each algorithm can be adapted for a specific field [10, 11].

- SHA family: Secure SHA Algorithms are a family of Hash Algorithms published by NIST since 1993.SHA has many derivative standards such as SHA-0, SHA-1, SHA-3
- MDA/MD5/MD6: Message-Digest Algorithm is a family of broadly used cryptographic hash function developed by Ronald Rivest that produces a 128-bit for MD4 and MD5, 256-bit for MD6
- Quark: Family of cryptographic functions designed for resource-constrained hardware environments.
- CubeHash: A very simple cryptographic hash function designed in University of Illinois at Chicago, Department of ComputerScience
- Photon: A lightweight hash – function designed for very constrained devices
- SQUASH : Not collision resistant, suitable for RFID applications

According to its complexity, Quark algorithms present the most appropriate to evaluate the performance of the soft core FPGA processor architecture. Quark can minimize area and power consumption, yet offers strong security guarantees. These Hash algorithms that are efficiently implemented in low cost embedded devices are important components for securing new applications in ubiquitous computing. Quark Hash algorithm It is a family of lightweight cryptographic "sponge" algorithms designed for resource-constrained hardware environments, as RFID tags. It combines a number of innovations that make it unique and optimized. In the design of Quark, designers opt for an algorithm based on bit shift. It combines a sponge construction with a capacity e equal to the digest length n, and a core permutation inspired by preceding primitives. Quark algorithm proposes three instances: u-Quark, d-Quark and s-Quark. These instances are parameterized by a rate r, a capacity c, an output length n and a b-bit permutation (b= r + c). Table 2 demonstrates the parameters of each instance of the Quark algorithms [10].

TABLE II. PARAMETERS OF QUARK HASH ALGORITHMS INSTANCE [10]

	Rate (r)	**Capacity (c)**	**With (b)**	**Digest (n)**
u-Quark	8	128	136	136
d-Quark	16	160	176	176
s-Quark	32	224	256	256

v. Using the Template

Processor performance can be measured in different metrics such as execution time, energy consumption and area utilization. The most common metric is the time required for a processor to accomplish defined task. In some architecture using internal CPU clock driver, execution time presents the clock driver multiplied by the total instruction cycle count. In our case, execution time is measured using a Logic Analyzer to have a high precision measurement.

A. Experimental Setup

Performance evaluation was estimated on a first time by a basic measurement of the different MicroBlaze soft core configurations implemented on Xilinx Virtex-5 development board (XUPV5-LX110T, xc5vlx110t, grade ff1136, speed -1). We used CoDevelopper tool to generate a co-processor using (HLS methodology) and the Xilinx Project Studio (XPS) for configuring the FPGA to include a MicroBlaze soft-core with a 64 KB (the maximum size possible), 125 MHz (the maximum frequency possible).

In our paper, we will automatically generate a cryptographic application as a co-processor added to a MicroBlaze with a FSL interface using Embedded Development Kit (EDK) which enables the integration of both Hardware and Software modules of an embedded system: The hardware architecture is first synthesized into a Gate-Level netlist, and then translated on the specific FPGA device. 1) The interconnections of these resources are, then, placed and routed. The downloadable .bit file is created for the whole architecture hardware design. 2) The software benchmarks are, then, compiled into an executable and linkable (ELF) file. The Microprocessor Software Specification (MSS) file and the Microprocessor Hardware Specification (MHS) file are used to describe software structure and hardware connection of the embedded architecture. EDK uses these two files to implement the design flow and finally assemble them into a single downloadable file ready to be implemented on FPGA.

International Conference on Control, Engineering & Information Technology (CEIT'14)
Proceedings - Copyright IPCO-2014, pp.146-151
ISSN 2356-5608

Fig. 3. Xilinx Virtex-5 FPGA Evaluation Platform

B. Evaluation Results

Performance was evaluated, in a first time, in terms of the number of Look Up Tables (LUTs) and Flip Flops (F-Fs) used. Table 3 presents the area consumption by exhaustively examining some possible MicroBlaze configurations.

TABLE III. THE AREA CONSUMPTION OF SOME MICROBLAZE CONFIGURATIONS

Configuration	With Optimization		Without Optimization	
MB + Units	**LUTs**	**LUTs**	**LUTs**	**LUTs**
Basic	1210	1657	1657	1693
BS	1570	1818	1818	1727
FPU	1620	2395	2395	2105
Mul	1456	1714	1714	1709
ID	1581	1801	1801	1805
MSRU	1458	1675	1675	1690
BS+mul+ID	1727	1964	1964	1867
BS+mul+FPU	2307	2511	2511	2162
BS+ID+FPU	2433	2668	2668	2258
BS+mul+ MSRU	1609	1830	1830	1749
BS+ID+ MSRU	1734	1966	1966	1846
BS+FPU+ MSRU	2313	2533	2533	2142
mul+ID+FPU	2358	2575	2575	2241
mul+ID+MSRU	1628	1864	1864	1829
mul+FPU+MSRU	2207	2418	2418	2127
ID+ MSRU+FPU	2359	2554	2554	2224
MSRU+FPU	2202	2417	2417	2107

Results of the area utilization of the different MicroBlaze soft core FPGA configurations prove that the average slices without using optimization option is very important. For each application, different MicroBlaze configurations can be provided and the resulting embedded system can be analysed in different metrics. The area consumption presents one of the metric in the choice of embedded processors systems. However, in real-time complex applications, both execution time and area consumption determine the efficiency and the high performance of the configured embedded soft core processor. In our paper, we compute the execution time of the configuration described on the table for the three Quark hash functions.

Quark hash functions don't use huge values. It is dominated by barrel shifter, integer arithmetic, logic decisions, and memory accesses intended to reflect the CPU activities in computing applications. It takes a huge time to memory access. To evaluate the performance of MicroBlaze soft core processor, we have to estimate the execution time in order to choose the efficient configuration which takes the minimum execution time onto the smaller hardware area. Fig. 4, 5 and 6 demonstrate the benefits of the Xilinx MicroBlaze soft-core for the Quark hash functions on 17 configurations.

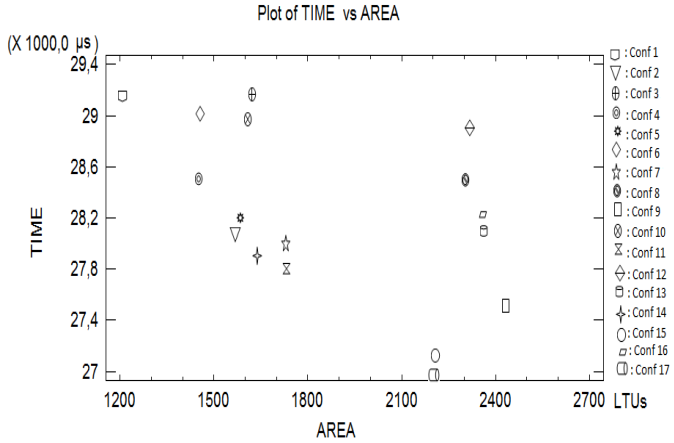

Fig. 4. The Rum Time and the area consumption of the s-Quark function

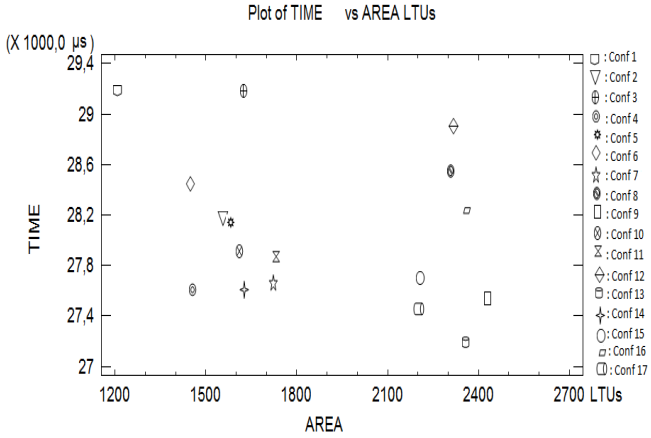

Fig. 5. The Rum Time and the area consumption of the s-Quark function

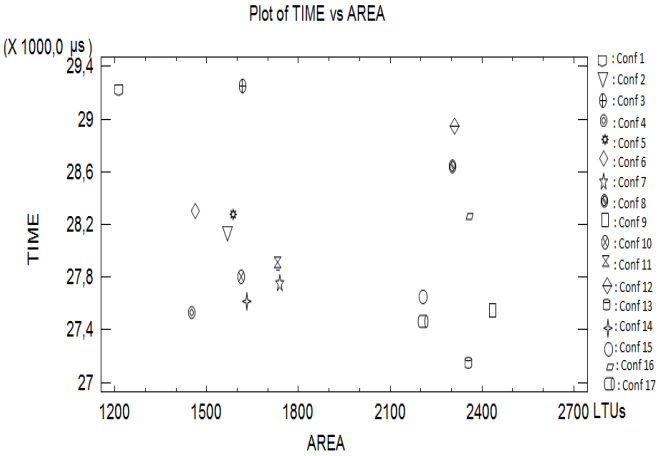

Fig. 6. The Rum Time and the area consumption of the d-Quark function

Figures demonstrate the benefits of configuring soft core processor in terms of execution time and area size constraints. Results prove that the efficient configuration of the MicroBlaze is the using of the barrel shifter and the machine status register units.

VI. Conculusion

FPGA soft cores present one of the attractive processors for implementing embedded applications. These soft cores, such as MicroBlaze, typically require longer execution times compared to the hard core processors, which reduce the number of potential application using soft core. However, they provide to designers the flexibility to be configured and enable those designers to rapidly build/implement and verify the FPGA designed processor.

The purpose of this paper was to show the effect of the MicroBlaze configuration on the execution time for a lightweight cryptographic application such as Quark algorithm Results demonstrate that the choice of the good configuration have a significant impact on the system performance. However, obtaining these results require approximately 20 minutes per configuration (60% of the time spent on synthesis). The same approach can be used to evaluate the performance of other embedded systems or other architectures.

References

[1] Xilinx, Inc. Xilinx Press Release #03142, http://www.Xilinx.com, (2013)

[2] Xilinx MicroBlaze processor reference guide embedded development kit EDK.7li, (2012).

[3] I. MHADHBI, S. BEN SAOUD: Model-Based Design approach for embedded Digital Controllers Design, International Journal of Automation and Control, 5. 267-283, (2011)

[4] N.LITAYEM, A.ACHBALLAH, I. MHADHBI, S. BEN SAOUD: Rapid Hardware-In-the-Loop Implementation for FPGA based embedded Controller for more reliable Electrical Traction Systems, Journal of Computer Science and Control Systems, 3. 51-58. (Octobre 2010)

[5] I. MHADHBI, A. ACHBALLAH, S. BEN SAOUD: High Level Synthesis Design Methodologies for Rapid Prototyping Digital Control Systems, 11th IFAC International Conference on Programmable Devices and Embedded Systems (2011), DOI: 10.3182/20120523-3-CZ-3015.00046

[6] Agarwal, A.: Comparaison of high level design methodologies for arithmetic IPs: Bluespec and C-based synthesis. Master of Science in Electrical Engineering and Computer Science. Massachusetts Institute of technology. (2009)

[7] Impulse Accelerated Technologies: www.implse.com, (2011)

[8] Lizy Kurian John: Performance evaluation: Techniques, tools and benchmarks. Electrical and computer engineering department, The University of Texas at Austin. (2009)

[9] I. MHADHBI, N. LITAYEM, S. BEN OTHMAN and S. BEN SAOUD: DSC Performance Evaluation and Exploration case of TMS320F28335, 02. 124-129. (2013)

[10] J. Aumasson, L. Hanzen, W. Meier, M. Naya-Plasencia: Quark: A Lightweight Hash. Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science, 6225. 1-15. (CHES 2010)

[11] Bogdanov, A., ESAT/SCD/COSIC, Leuven, Knezevic, M., Leander, G., Toz, D.: SPONGENT: The Design Space of Lightweight Cryptographic Hashing. Computer IEEE Transaction, 62. 2041-2053. (Oct. 2013)