

## Platform Selection According to SoC Architecture: a case study

Y. Aoudni<sup>1,2</sup>, N. Ben Amor<sup>1,2</sup>, G. Gogniat<sup>1</sup>, J.L. Philippe<sup>1</sup>, M. Abid<sup>2</sup>

<sup>1</sup>LESTER, Université de Bretagne Sud CNRS FRE 2734, Lorient, France,

<sup>2</sup>GMS, ENIS engineering school, Sfax, Tunisia

Email : aoudni@iuplo.univ-ubs.fr

### Abstract

*This work aims to compare several tools and SoC platforms according to the following key parameters: FPGA architecture, coprocessor and accelerator integration, RTOS and HW-SW refinement tools. These key parameters are required to select a flexible and efficient SoC Platform (and the associated tools) in order to implement an efficient PACM (Processor – Accelerator – Coprocessor – Memory) architecture model. A case study for the PACM architecture model has been targeted in order to validate proposed key parameters. Four Platforms were candidate to implement the PACM architecture model. Finally, a Nios development kit (Quartus, SOPC Builder and STRATIX device) is used as a prototyping platform for PACM architecture and a shading algorithm for 3D image treatment was implemented as an application to prove the SoC platform adequacy with our PACM architecture model.*

### 1- Introduction

The designs of SoC in many application domains are often subject to stringent requirements in terms of processing performance and flexibility [1]. To enable flexible low-cost designs in a short design cycle, emerging designs are based on hardware/software SoC platforms that integrate multiple software (SW) programmable resources, e.g. DSP and microcontroller cores, together with dedicated hardware (HW) accelerators within a single cost-efficient chip. Programmability is introduced in these single-chip architectures (thus offering the desired flexibility in the design), while maintaining most of the advantages of customized VLSI solutions (such as the potential to optimize the processing performance and power dissipation) [2].

Hardware/software co-design method exists today for designing the different hardware and software resources of a SoC architecture [3]. As time to market pressures and product complexities climb, the need to reuse complex building (also known as Intellectual Property (IP) or Virtual Component (VC)) also increases. These components represent functions for specific domain like signal processing (DCT, FFT), telecommunication and multimedia (VLC, Turbo codes) etc. In this area, SoC

implementation management requires a robust co-design environment in order to master the complexity and the different refinement steps of the system from a high level specification.

Depending on the application, different application-specific architectures using different execution models and combinations of software and hardware components may be required (e.g. driven by programmability, performance, and power computing requirements). Even the choice of the

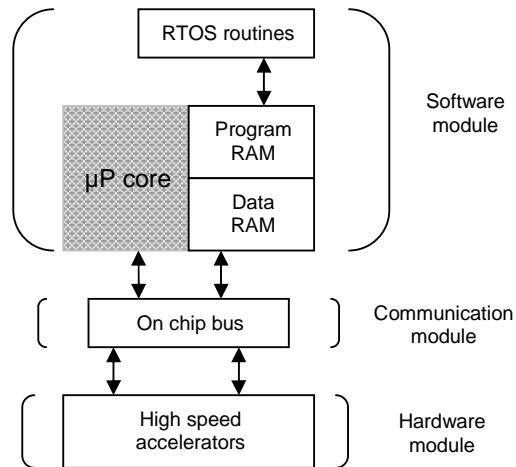


Figure 1: reactive/dataflow SoC architecture model

programmable processor to use is heavily dependent on the application. Briefly, the question to answer is: can the same CAD tools and prototyping platforms address all architecture models or is there a trade-off between these platforms and architecture models? In this paper we propose an answer and demonstrate it using our case study.

In literature we find many proposed architecture for digital applications that can be classified with different granularity levels: finite, coarse and multi-grain. For example, the FPGA families represents the finite grain level: Altera Stratix II, Xilinx Virtex II, etc. Systolic Ring [4], DART [5], RAW [6] and KressArray[7] beyond to the coarse grain level. And for multi-grain level we find GARP [8], Chameleon [9], etc. Each of these architectures targets a specific domain like DART for mobile application, Systolic Ring and KressArray for data flow application. But in reactive/dataflow system

we need another vision for the system architecture. As a solution we proposed in figure 1 SoC architecture model for reactive/data flow system. This is the model we have selected to implement the PACM (Processor – Accelerator – Coprocessor – Memory) architecture model as will be explained in the following.

This paper presents a case study comparison of several CAD tools and reconfigurable SoC prototyping platforms to select the suitable ones for the target architecture (see figure 1). Firstly, we explain the new main features for SoC co-design environment. Then we present the comparative study and the key parameters of selection extracted from the PACM architecture model. A 3D shading algorithm example has been implemented to demonstrate the SoC platform adequacy with our PACM architecture model. Finally, we end with conclusion.

## 2- SoC co-design environment

Figure 2 describes a typical SoC co-design environment. Starting from a high level system model entry a HW/SW partitioning tool to select the hardware (HW) or the software (SW)

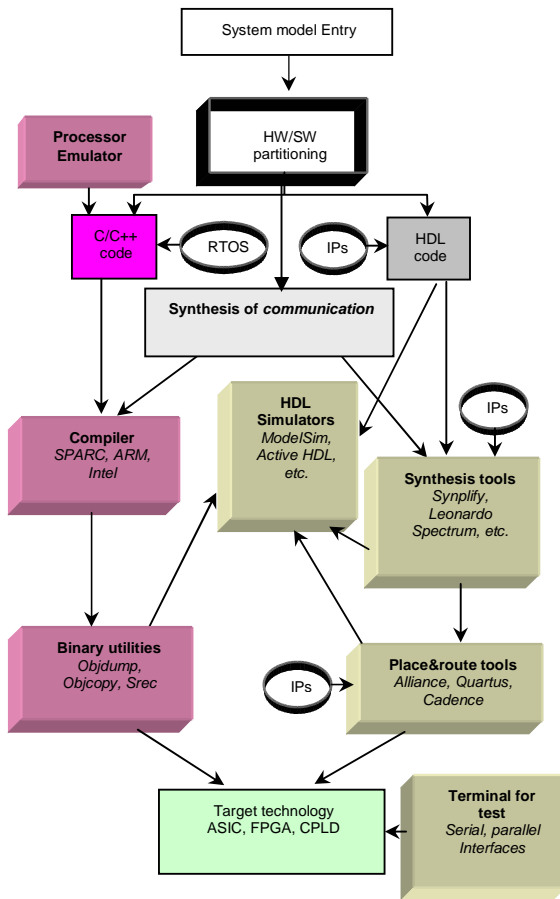


Figure 2: SoC Environment Design

implementation of the functions composing the system. Then, each function is treated by the appropriate refinement environment. The two next subsections discuss the HW and SW refinement environments presented with clear (HW) and dark (SW) grey color in figure 2. Then we present the synthesis of communication.

### 2-1 Software refinement environment

If we analyze needs concerning the development of the SW parts, we can do the following reports. In fact the environment of software refinement allows to build executable codes from a specification described with high level languages (e.g. C language, C ++, Java). So, the generation of binary codes is made by means of compilers. Generally, a compiler is specific to a language of high level description and a model of execution. Basically, a SoC based on integrates a processor core (ARM, SPARC, Power PC...) [10] requires a specific compiler. Moreover, an emulator should be used as a tool to verify the execution of SW functions at a high level abstraction before any implementation. In addition, new embedded reactive systems need to implement a scalable RTOS (Real Time Operating System), so the software environment must support this new option which will be used for the management of the memory, the access time to the tasks, etc. Examples of RTOS are Embedded Linux, MicroC/OS-II, RTEMS, etc. In practice, if a RTOS is present it is necessary to port it to the processor core and to generate a scalable operating code [11]. Then, programs generated by compilers must be downloaded into the target electronic modules (RAM, ROM, flash memory). Thus it is necessary to have binary utilities which allow the designer to build specific binary codes for the target electronic devices. These binary utilities such as Objdump, Objcopy must be able to generate also data files needed for co-simulation with HDL simulator like ModelSim [12].

### 2-2 Hardware refinement environment

If we analyze needs concerning the design of the HW parts, we can do the following reports. The hardware refinement environment contains the following tools: synthesis, place&route as well as HDL simulation tools. As the design of SoC corresponds to complex applications in terms of architecture and functions, it is necessary to have a design environment able to support this complexity. For that reason, researchers in the SoC field turn to the use of new generation of commercial HDL simulation tools (ModelSim, NC-Verilog, NC-VHDL, Synopsys, VCS, Verilog-XL, VSS), synthesis tools (Leonardo Spectrum, Synplify Pro, Express Synopsys FPGA Tools) and place&route tools (Alliance, Quartus, Cadence). In [12] we

presented the benefits of using design tools for simulation and synthesis to analyze the performances of SoC in terms of quality, computing time, allocated area, etc. Furthermore, these refinement tools target a particular range of FPGA families depending on the vendors. For example we find a specific HDL simulation tool for the Altera FPGA family (ModelSim-Altera) or Xilinx FPGA family (ModelSim Xilinx Edition).

The new features added to HW refinement environment are the use of IP model libraries at different abstraction levels and the HW/SW on chip communication. To be able to re-use IPs, CAD tools vendors provide the designer IP models library. But generally the designer must also specify some parts of the SoC or adapt the IP models to his specific SoC [13].

### 2-3 Synthesis of communication

Synthesis communication task aims to identify and generate the communication protocols between HW and SW modules [14]. For the HW/SW communication, we find new type of resources like on chip buses, internal networks, etc. These new communication resources use communication protocols that are not standardized yet [15], what complicates the HW/SW interface in SoC design.

All these new aspects of IPs re-use, SoC communication synthesis and RTOS scalability lead us to use a SoC co-design environment able to support all these new features. In addition, the CAO tools and platforms considered within the SoC co-design environment must be compatible with the SoC architecture model. This point is explained in next the section where we present the PACM architecture model.

### 3- SoC co-design environment for the PACM architecture model

In this section, for the architecture presented in figure 3 and based on the model presented in figure

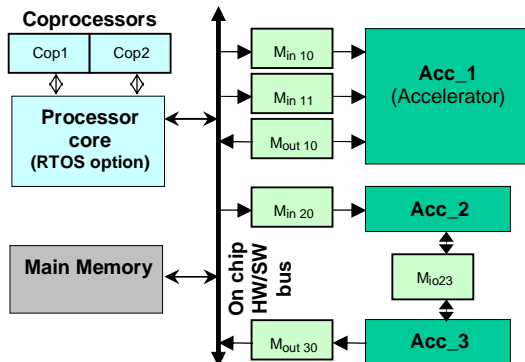


Figure 3: PACM architecture model

1, we show that several CAO tools and platforms, similar in first view, do not present the same adequacy degree with the targeted model of execution. The figure 3 presents the PACM model. Basically our architecture is built around a processor core (for example Nios, ARM, LEON...) which offers configuration opportunities for adding coprocessors reached through the main processor registers (for example floating point unit, HW divider, HW mathematic functions ...).

The processor communicates with dedicated HW accelerators through a standard on chip HW/SW bus (e.g. Amba, Avalon, IBM CoreConnect...) using control logic and specific memory blocks. Coprocessors and HW accelerators usage depends on the application complexity and on the computing constraint requirements. In order to give more flexibility and adaptability to the SoC, we have chosen the reconfigurable technology to implement our SoC.

If we analyze the key parameters of the PACM architecture model, the adequate platform must integrate the following features:

- The platform must integrate a FPGA device characterized by a heterogeneous architecture (logic elements, DSP blocks, RAM blocks, I/O pin...) and by a size able to integrate the HW and SW parts of the SoC.
- The platform must provide a processor core that gives opportunities to integrate some coprocessors within its ALU and reached through the processor main registers to get an ASIP model.
- The HW accelerators integration must be supported using an on chip HW/SW bus or other on chip HW/SW communication module.
- RTOS option with the corresponding port to the targeted processor core must be present.
- The HW and SW refinement tools must be robust and efficient to limit the time-to-market constraint.

All these key parameters correspond to the criteria to select a suitable SoC platform. We made

Platforms	LEON	Nios kits	Excalibur kit	PowerPC Microblaze kits
Key parameters				
FPGAs architecture	Xilinx family >= Virtex	Altera family >=APEX	APEX family only	Xilinx family>Virtex
Coproductors integration	+	+++	+++	----
Accelerator integration	+	+++	+++	+++
RTOS	++	+++	+++	+++
SW and HW refinement tools	+++	+++	+++	+++

Table 1: Platforms comparison

a qualitative study for different representative platforms, and evaluate their adequacy with the PACM architecture model.

We performed an experimental study based on the main features of SoC platforms. The results are presented in table 1. We notice that all presented SoC platforms provide a robust and efficient HW and SW refinement tools like ISE Xilinx tool and Quartus, on chip HW/SW bus as AMBA (LEON and Excalibur kits) and IBM CoreConnect (PowerPC and Microblaze kit) and also a port for many RTOS like RTEMS ported on LEON and ARM, WindRiver port on PowerPC and Microblaze, etc. However, only Nios, ARM and LEON cores can support coprocessor feature. In addition, coprocessor integration in Nios and ARM cores is more rapid and flexible using the virtualization and custom instruction generation given by SOPC Builder tool. Also, Nios SoC can be implemented in large STRATIX family [16] which contains DSP blocks and different sizes of RAM blocks, unlike ARM development kit which is restricted to APEX device and its core is a hard IP and not a soft one like the Nios core. Thus, we notice that the SoC platform based on Nios processor core kit [17] provided with Quartus and SOPC Builder environments by Altera [18] is the most suitable to design a reconfigurable SoC using the PACM architecture model. Indeed, SOPC Builder tool gives the designer a virtual image of the Nios processor soft core and the accelerators can be linked to the Nios processor core through the Avalon on chip bus. Custom instructions are also provided with this platform in order to facilitate the coprocessors integration within the Nios ALU. Namely, our choice is based on this last feature in order to implement a reconfigurable ASIP core.

As a conclusion of this analysis we can see that the available CAD tools and SoC platforms can not address all the architecture models and that a study must be done in order to select the suitable platform for the appropriate architecture model. In our case the Nios processor core kit is suitable to the PACM architecture model.

In the next section, to illustrate the efficiency of using the Nios processor core kit to implement the PACM architecture model we present a case study on shading algorithms for 3D image treatment. Our platform is composed of Quartus and SOPC Builder environments, Nios soft core processor, a EP1S40 STRATIX device and the MicroC/OS-II RTOS [19]. Note that the RTOS implementation will be done in future work.

#### 4 – Application

#### 4-1 LAMBERT and GOURAUD algorithms

In this case study, we have implemented two 3D shading algorithms used in image synthesis: LAMBERT and GOURAUD [20]. These algorithms consist to cut an object in a number of polygons and then to determine the intensity of the light in every polygon of the object. Several mathematic expressions and vectors analysis must be done in order to get a light intensity value for one polygon. The basic operations of these computing operations are the addition, subtraction, multiplication and square root. Indeed, for each polygon we must calculate the area normal vector, light direction vector and the angle between these two vectors. The difference between the two algorithms is that GOURAUD takes into account the continuity between neighbor polygons unlike LAMBERT (see figure 4a and 4b).

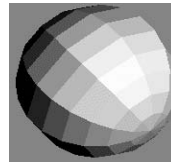


Figure 4a: shading with LAMBERT algorithm

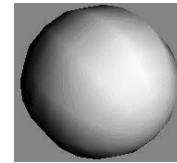


Figure 4b: shading with GOURAUD algorithm

#### 4-2 Implementation

LAMBERT and GOURAUD algorithms can be implemented in the Nios as a SW program. But in order to accelerate the execution a solution consists to use HW accelerators. Another solution can be done by extracting coprocessors from the SW execution in order to speedup some critical parts of the code. In general these critical parts correspond to nested loop body operations that are executed a large number of time. These two types of implementation are presented in the following paragraphs right after the full SW algorithms implementation.

Firstly, we implemented these two algorithms as two accelerators linked to the Avalon on chip bus. Indeed in the Quartus environment, we can design a HW accelerator using a Block Design File (BDF) as presented in figure 5, then we can generate the specific VHDL or Verilog code and finally we can run the compilation to simulate and to get the bitstream file. The BDF feature in Quartus environment helps the designer to get a right HDL code in less time with optimized components provided within the vendor library. We have used the PIO interface feature in SOPC Builder to link the HW accelerators with the Avalon bus. This tool is able to generate automatically the interface

between the HW and the SW components that is helpful to reduce the HW/SW SoC time design.

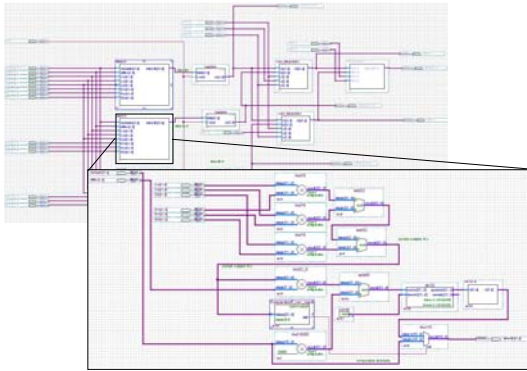


Figure 5: LAMBERT and GOURAUD Block Design Files

Secondly, we integrated mathematic coprocessors as custom instructions within the Nios soft core. In this case, the virtual design entry of the Nios in SOPC Builder is very helpful and efficient to configure the processor core, especially for adding either 16 or 32 bits coprocessors via two main ALU registers. The key point in this feature is the generation of specific coprocessor custom instruction known by the Nios C compiler.

Table 2 gives the results obtained for the LAMBERT and GOURAUD algorithms using the accelerators and the coprocessors. The results are given in term of required logic elements and in term of execution time speedup compared to a full software implementation using the Nios. We can notice that the speedup obtained using GOURAUD accelerator is more important than the coprocessor one, but it needs more area (i.e. FPGA resources). On the other hand coprocessors LAMBERT implementation is more benefit than accelerator ones in terms of logic cell and speedup. Thus, the

		Logic cell	DSP block	RAM Block (bits)	Speedup
Nios32 Core		6164	0	669696	-
Nios with accelerators	lamBERT	922	8	4096	10,8%
	gouraUD	4201	28	12288	55,3%
Nios with coprocessors	Adder32	32	0	0	68,9% (lamBERT)
	Sub32	34	0	0	
	Mult32	0	8	0	11,3% (gouraUD)
	Sqrt32	21	0	0	

Table 2: Results obtained for the LAMBERT and GOURAUD algorithms

selection between accelerators or coprocessors implementation depends on the SoC execution time and FPGA resources constraints. Note that using this SoC platform a few workdays are just needed for an effective implementation of these two complex algorithms which prove the efficiency of

the key parameters to select a SoC platform for the PACM architecture model.

## 5- Conclusion

In this paper we have presented the new features of SoC platform and co-design environment and the main role of hardware and software refinement tools to implement SoC especially on FPGA device. Based on the PACM architecture model, we have shown that the available CAO tools and platforms can not address all the SoC architecture models and an analysis must be done to select the suitable platform for the appropriate architecture model. The key parameters that we have considered to select the right SoC platform are the following ones: FPGA architecture, coprocessor and accelerator integration, RTOS and HW-SW refinement tools. Based on these parameters a platform based on the Nios STRATIX development kit was chosen to implement the PACM architecture model. To validate our choice two 3D shading algorithms were implemented on the Nios platform using the Quartus and SOPC Builder. Results have demonstrated the benefits using such platform for the PACM architecture model. In future work, we propose to implement the RTOS subroutine and to study the SoC power performance for the PACM architecture with video compression algorithms.

## References

- [1] Julio C, B.Mattos, Luigi Carro, Efficient architecture for FPGA-based microcontrollers, ISCAS paper number 2825, 2002.
- [2] Amit Singh, Malgorzata Marker-Sadowska, Efficient circuit clustering for area and power reduction in FPGAs, SIGDA 2002.
- [3] Bill Lin, Karl Van Rompaey, Stenven Vercauteren, Designing single chip systems, ASIC 1996.
- [4] G. Sassatelli, L. Torres, P. Benoit, T. Gil, C. Diou, G. Cambon, J. Galy, Highly Scalable Dynamically Reconfigurable Systolic Ring-Architecture for DSP Applications. In IEEE Design Automation and Test in Europe Paris 2002, pp. 553-557, DATE 202.
- [5] Raphaël David, Daniel Chillet, Sebastien Pillement, and Olivier Sentieys. Dart : A dynamically reconfigurable architecture dealing with next generation telecommunications constraints. In 9th IEEE Reconfigurable Architecture Workshop RAW. IEEE CS Press, April 2002.
- [6] Waingold, E.; et al. Baring it all to software: Raw machines. Computer, vol. 30(9), pp. 86-93, 1997.
- [7] R. Kress et al. A Datapath Synthesis System for the Reconfigurable Datapath Architecture. In Asia and South Pacific Design Conference, Asp-DAC 95.
- [8] Callahan, T.J.; Hauser, J.R.; Wawrzynek, J. The Garp architecture and C compiler. Computer, vol. 33(4), pp. 62-69, 2000.

- [9] B. Salefski, L. Caglar. Re-Configurable Computing in Wireless. In 38th Design Automation Conference, DAC 2001, Louisiana, USA, June, 2001.
- [10] F. Campi, A. Cappelli, A. La Rosa, L. Lavagno, R. Canegallo, A Reconfigurable Processor Architecture and Software Development Environment for Embedded Systems, Reconfigurable Architecture Workshop, Nice France RAW'2003.
- [11] V. Nollet, P. Coene, D. Verkest, S. Vernalde, R. Lauwereins, Designing an Operating System for a Heterogeneous Reconfigurable SoC, Reconfigurable Architecture Workshop, Nice France RAW'2003.
- [12] Y. Aoudni, I. Maalej and al., Analysis of hardware/software System on Chip: Case Study, IEEE Conference on Signal, Systems, Decision and information theory, Sousse, Tunisia, 26-28 March, SSD'2003.
- [13] M. Bolado, H. Posadas and All, Platform based on Open-Source Cores for Industrial Applications, Design Automation and Test in Europe February 16-20, Paris DATE'2004.
- [14] I. Maalej, G. Gogniat, M. Abid, J-L Philippe, 'Generic bus communication for system on chip communication Design', SSD'03 26-28 Mars, 2003 Sousse, Tunisie.
- [15] Vesa Lahtinen, Kimmo Kuusilinna, Tero Kangas, Timo Hamalainen, Interconnection scheme for continuous-media systems-on-chip, pages 123-138 Microprocessors and Microsystems 26, 2002.
- [16] Altera web site STRATIX family data sheet, April 2004  
[http://www.altera.com/literature/hb/stx/stratix\\_section\\_1\\_vol\\_1.pdf](http://www.altera.com/literature/hb/stx/stratix_section_1_vol_1.pdf)
- [17] Altera web site, Nios Development Kit STRATIX Edition, Getting Started User Guide July 2003  
[http://www.altera.com/literature/ug/ug\\_nios\\_gsg\\_stratix\\_1s10.pdf](http://www.altera.com/literature/ug/ug_nios_gsg_stratix_1s10.pdf)
- [18] Altera web site, SOPC Builder User Guide, June 2003  
[http://www.altera.com/literature/ug/ug\\_sopcbuilder.pdf](http://www.altera.com/literature/ug/ug_sopcbuilder.pdf)
- [19] Jean J. Labrosse, MicroC/OS-II The Real Time Kernel Second Edition, CMP Books 2002.
- [20] Tomas Akenine Moller, REAL-TIME RENDERING second edition page 70, AK Peters Ltd 2002.