IP/SOC 2004

Session: ASIC DESIGN PLATFORM

IP Processor Core Platform Selection According to SoC Architecture: a case study

Aoudni Yassine, GMS/LESTER Laboratories

Sfax/Lorient, Tunisia/France

Abstract:

This work aims to compare several IP core processor based 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 high performance IP processor core (and the associated tools) in order to implement an efficient mono-processor PACM (Processor – Accelerator – Coprocessor – Memory) architecture model. A case study for the PACM architecture model has been targeted in order to validate our key parameters. Four IP processor cores were candidate to implement the PACM architecture model. Finally, the selected ones corresponds to the Nios soft processor core within its development kit (Quartus, SOPC Builder and STRATIX device) and a shading algorithm for 3D image treatment was implemented to prove the IP processor core platform adequacy with our PACM architecture model.

TITLE: IP Processor Core Platform Selection According to SoC Architecture: a case study

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 processor cores as programmable resources 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 processor architectures (RISC, SPARC…) and 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 programmable processor to use is heavily dependent on the application. Briefly, the question to answer is: can the same IP processor core prototyping platforms [4] 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.

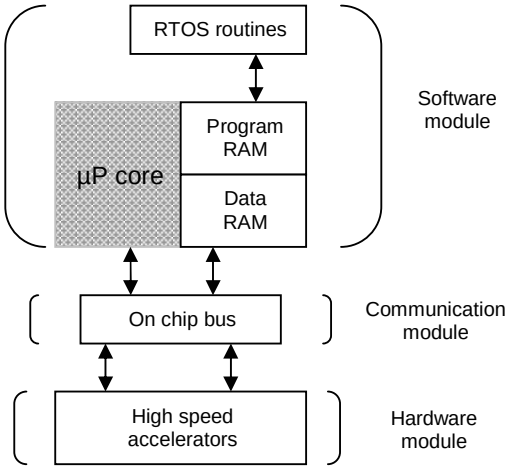

Figure 1: SoC architecture model based on a

A classical mono-processor SoC architecture model is depicted in figure 1. 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 IP processor cores prototyping platforms to select the suitable ones for the target architecture (see figure 1). Firstly, we present the comparative study between IP processor cores based platforms and the key parameters of selection extracted from the PACM architecture model. A 3D shading algorithm example has been implemented to demonstrate the IP processor core platform adequacy with our PACM architecture model. Finally, we end with conclusion.

2. IP processor core platform for the PACM architecture model

In this section, for the architecture presented in figure 2 and based on the model presented in figure 1, we show that several IP processor cores 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 ...).

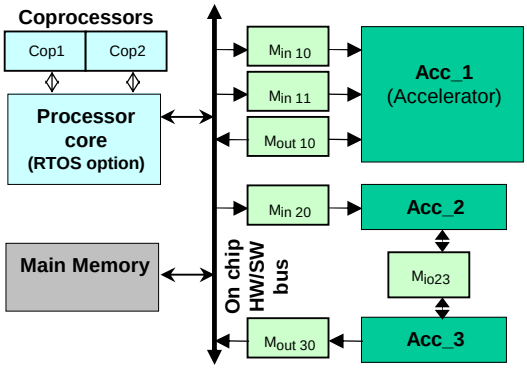

Figure 2: PACM architecture model

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 IP processor core platform must integrate the following features:

- The IP core processor must be implemented within 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 IP processor core must gives opportunities to integrate some coprocessors within its ALU and reached through the processor main registers to get an ASIP model [5].
- The IP processor core and HW accelerators must be linked by an on chip HW/SW bus or other on chip HW/SW communication module [6].
- RTOS option with the corresponding port to the targeted IP processor core must be present [7].
- The HW and SW refinement tools must be robust and efficient to limit the time-to-market constraint [8].

All these key parameters correspond to the criteria to select a suitable IP processor core platform. We made a qualitative study for different representative IP processor core platforms, and evaluate their adequacy with the PACM architecture model.

IP processor platform / Key parameters	LEON	Nios kits	ARM Excalibur kit	PowerPC Microblaze kits
FPGAs architecture	Xilinx family >= Virtex	Altera family >=APEX	APEX family only	Xilinx family>Virtex
Coprocessors integration	+	+++	+++	----
Accelerator integration	+	+++	+++	+++
RTOS	++	+++	+++	+++
SW and HW refinement tools	+++	+++	+++	+++

Table 1 : IP processor core platforms comparaison

We performed an experimental study based on the main features of IP processor core platforms. The results are presented in table 1. We notice that all presented IP processor core 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 ARM Excalibur kits)

and IBM CoreConnect (PowerPC and Microblaze kits) and also a port for many RTOS like RTEMS port on LEON and ARM, WindRiver port on PowerPC and Microblaze, etc. However, only Nios, ARM and LEON IP cores can support coprocessor feature. In addition, coprocessor integration in Nios and ARM IP cores is more rapid and flexible using the virtualization and custom instruction generation given by SOPC Builder user-friend tool. Also, Nios SoC can be implemented in large STRATIX family [9] 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 IP processor core [10] provided with Quartus and SOPC Builder environments by Altera [11] is the most suitable to design a reconfigurable SoC based on IP processor core 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 IP processor cores platforms can not address all the SoC architecture models and that a study must be done in order to select the suitable IP processor core 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, an EP1S40 STRATIX device and the MicroC/OS-II RTOS [12]. Note that the RTOS implementation will be done in future work.

3. Experimentation case study

In this case study, we have implemented two 3D shading algorithms used in image synthesis: LAMBERT and GOURAUD [13]. 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 neighbours polygons unlike LAMBERT (see figure 3a and 3b).

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

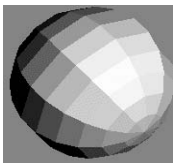

Figure 3a: shading with LAMBERT algorithm

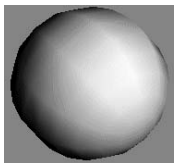

Figure 4b: shading with GOURAUD algorithm

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 4, then we can generate the

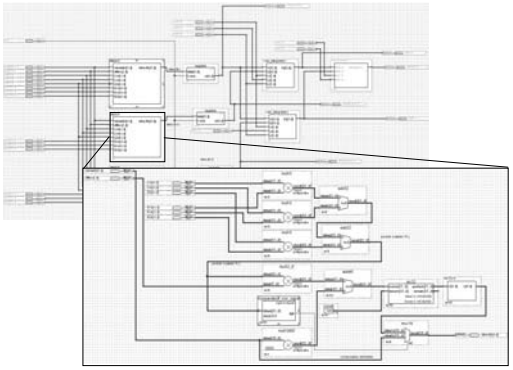

Figure 4: LAMBERT and GOURAUD Block Design Files

specific VHDL or Verilog code and finally we can run the compilation to simulate and to get the bit stream 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.

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 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.

		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) 11,3% (gouraud)
	Sub32	34	0	0	
	Mult32	0	8	0	
	Sqrt32	21	0	0	

Table 2: Results obtained for the LAMBERT and GOURAUD algorithms

4. Conclusion

In this paper and based on the PACM architecture model, we have shown that the available IP processor cores platforms can not address all the SoC architecture models and an analysis must be done to select the suitable IP processor core platform for the appropriate architecture model. The key parameters that we have considered to select the right IP processor core 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, Malgorazta 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] M. Bolado1, H. Posadas1and All, Platform based on Open-Source Cores for Industrial Applications, Design Automation and Test in Europe February 16-20, Paris DATE'2004.

[5] 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.

[6] Vesa Lahtinen, Kimmo Kuusilinna, Tero Kangas, Timo Hamalainen, Interconnection scheme for continuous-media systems-on-chip, pages 123-138 Microprocessors and Microsystems 26, 2002.

[7] 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.

[8] Y .Aoudni, I. Maalej and al., Analysis of hard ware/software System on Chip: Case Study, IEEE Conference on Signal, Systems, Decision and information theory, Sousse, Tunisia, 26-28 Marsh, SSD'2003.

[9] Altera web site STRATIX family data sheet, April 2004

http://www.altera.com/literature/hb/stx/stratix_section_1_vol_1.pdf

[10] Altera web site, Nios Developpement Kit STRATIX Edition, Getting Started User Guide July 2003

http://www.altera.com/literature/ug/ug_nios_gsg_stratix_1s10.pdf

[11] Altera web site, SOPC Builder User Guide, June 2003

http://www.altera.com/literature/ug/ug_sopcbuilder.pdf

[12] Jean J. Labrosse, MicroC/OS-II The Real Time Kernel Second Edition, CMP Books 2002.

[13] Tomas Akenine Moller, REAL-TIME RENDERING second edition page 70, AK Peters Ltd 2002.