Algorithm-Architecture matching, an application to the phase diversity algorithm

Romain BRILLU[1,2], Sébastien PILLEMENT[1], Fabrice LEMONNIER[2], Philippe MILLET[2], Marc BERNOT[3], Frédéric FALZON[3]

[1]LUNAM Université, Université de Nantes, IETR UMR 6164, Rue Christian Pauc, 44306 Nantes, FRANCE

[2]Thales Research & Technology Campus Polytechnique, 1 avenue Augustin Fresnel, 91767,Palaiseau,France

[3]Thales Alenia Space France, 100 Boulevard du Midi, 06156 Cannes la Bocca, France

{firstname.lastname}@thalesgroup.com, Sebastien.Pillement@univ-nantes.fr,
{firstname.lastname}@thalesaleniaspace.com

Abstract – The deployment of an application onto an architecture is often a long and difficult process. This is due to the fact that the characteristics of the target architecture is often taken into account late in the development process of an algorithm which makes the adequacy between architecture and algorithm more and more difficult.We propose in this paper a formalization of the design process which allows interactively, the calculation of metrics, the realization of simulation, thus in order to guide the transformations to be operated on the algorithm or on the architecture to optimize the global performance.The approach is exposed for a space application, which requires both a high computing power and an architecture compliant with the space constraints. Our methodology based on an algorithm description and a library of architecture determines the most appropriate algorithm and architectural implementation. This methodology has been tested for real cases implementation, and has shown promising results.

1 Introduction

The hardware architectures for image or signal processing applications have seen a rapid evolution. These platforms are now constructed by assembling standard processors, programmable specific circuit (Digital Signal Processor "DSP", Field Programmable Gate Array "FPGA") and integrated circuit (Application Specific Integrated Circuit "ASIC") [11].

This evolution associated with improved methods and tools for decision support, have allowed to achieve at reasonable cost implementations of complex algorithms that we would not have been considered a few years ago. We gradually evolved from a separate study of algorithms and architectures, to an overall methodological approach more and more formalized [6].

The Algorithm Architecture Matching ("AAM") is the simultaneous study of algorithmic and architectural aspects of an application by taking into account their interactions. Thus in order to obtain the most optimized algorithm implantation on a well-defined hardware platforms. The AAM allows on one hand to perform formal verification, to ensure the design continuity and, to pose optimization problems allowing the best architectures design.

However, since the nature of the architecture and the nature of the application constraints may vary over time, the difficulty of controlling the interaction between architecture and algorithm finds greatly increased. The AAM is then a reciprocal matching process of the algorithm and the architecture. It shall be based on a system-level formulation of the algorithms and the architectures, taking into account, the constraints (real-time, power consumption, environmental...), the computing power requirements, the nature of the information to be processed (data-flow, address based) and the nature of the operation to be done. The optimized configuration of the architecture and the algorithm then involves many complex research issues and spotlight the need of a clearly defined methodology allowing a fast and efficient design space exploration.

This is why we propose a platform-based design methodology which is based on the usage of formal modeling techniques, clearly defined abstraction levels and the separation of concerns to enable an effective design process. Thus in order to ease the interaction between the application and the architecture, to efficiently reduce the time to market and to prune solution space by providing design assistance.

The rest of this paper is organized as follows section 2 present our methodology, the phase diversity applications and the architectural constraints related to the spatial domain are presented into section 3, the results are given into section 4. Finally, conclusions and perspectives are drawn into section 5.

2 Design space exploration

The methodology (Figure 1) that we propose is based on SpearDE and OVP tools, as well as a C code and a library of components provided as input.

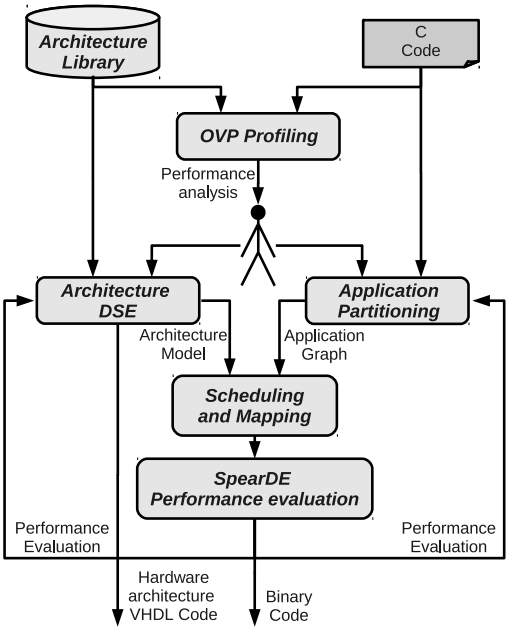

FIGURE 1 – Representation of the work flow used for our methodology

2.1 Tools presentation

2.1.1 SpearDE

SpearDE [10], software environment developed by Thales Research & Technology, allows the implementation of signal or image processing applications on "MPSoC" architectures. It is designed to support an implementation flow (Figure 2)

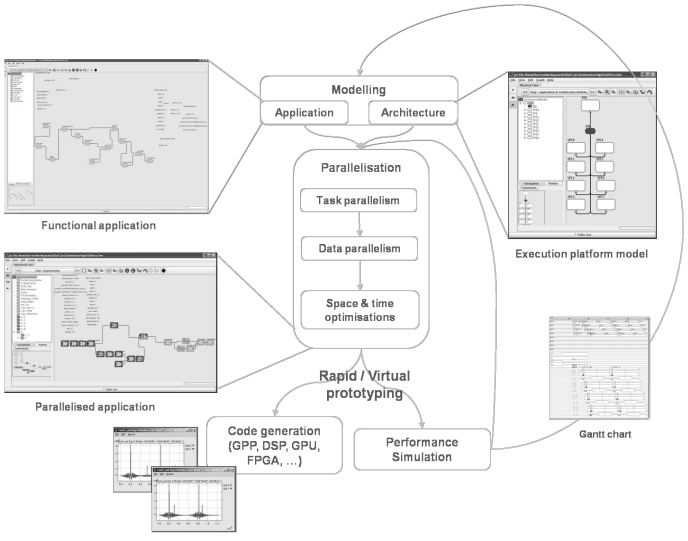

FIGURE 2 – SpearDE work flow representation.

of an application on an architecture written using a specific model. The application model, express the algorithm through a synchronous data-flow graph ("SDFG") that makes parallelism properties explicit. The architecture model highlighting, the structure of the target machine under an appropriate description. From these models and with the help of the tool, the matching of the application and the architecture is realized by defining a mapping of the application on the architecture. Once the placement obtained, SpearDE allows on one hand to trigger code generators in order to provide a parallel code executable on the targeted machine and secondly to generate a SystemC simulator to evaluate the performance of the various target architectures that can run the application.

SpearDE will then have a key role in our work flow because it will allow to get performance estimations. However, this tool can no be used as a unique solution, since it does not provide any information about the potential application and architecture bottleneck and about the application inherent parallelism, which are necessary information when you want to construct jointly the application and the architecture.

2.1.2 OVP

OVP [2] framework developed by Imperas company propose to model hardware architectures ranging from the simple processor to Mutiple Processor System on Chip ("MPSoC") through the use of the OVP simulation platform (Figure 3). The

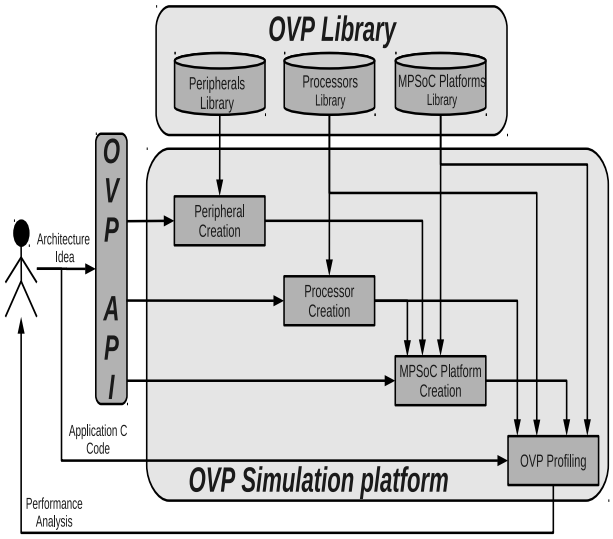

FIGURE 3 – Representation of the possibilities provided by OVP tool.

OVP environment seek to answer three types of distinct needs : 1) the validation and estimation of performance of an application, 2) the description and validation of hardware devices, and 3) it allows to model and simulate MPSoC architectures. To that end this tool is build around three axis : (1) a simulation platform OVPSim, (2) a set of application programming interface ("API"), and a rich library of models composed of both processors, peripherals and MPSoC platform.

In the context of our work flow, OVP is used to determine for each tasks of the application and for each processing cores

present into the library, the number of instruction needed by each task to be executed onto each core. From this number it is easy to get the computation time of the task onto the processing core.

Since OVP does not provide any performance estimation the role of this tool is then limited to provide to the user profiling information in order to identify the application and architecture bottlenecks.

2.2 Work flow for the DSE

As depicted on (Figure 1) based on these tools and on the furnished inputs (C code and a library of components), the first step of our work flow is to realize a C code profiling through the use of OVP tool, this in order to determine which are the requirements of the application, the potential parallelism and the application and architecture bottlenecks.

From these results, the user can either choose, to redefine the architecture or the targeted architecture, adapt the algorithm if the demands in terms of resources (memory bandwidth, computing power...) are too important or carry out these actions jointly. Once the application graph and architecture model determined, the application is then mapped on the architecture.

Based on this mapping SpearDE generates the scheduling of the application onto the architecture and a SystemC simulator which provide performance evaluation. From these evaluations the user can then choose to keep the solution (if it is not improvable or if it meets the expressed need), or the user can also modify the architecture or the application in consequence.

If the solution is chosen SpearDE will then be triggered to generate the executable binary code, while the user will generate the VHDL code by hand.

3 Phase diversity application

Controlling the stability of a space optical telescope requires the precise knowledge of the wavefront error caused by the misalignments of optical parts. Phase retrieval for in orbit space telescope characterization occurred for the first time when the Hubble space telescope showed critical optical aberrations [3, 5, 4] due to errors in the primary mirror figure. The methods developped to solve this problem are very well suited to the retrieval of telescope aberrations through the instrument Point Spread Function (PSF) computed from the observation of point objects . They are, however, not able to infer optical aberrations from extended objects or scenes. In the Earth observation domain, the telescope observes only extended scenes and the challenge is thus to compute the telescope wavefront error not from stars but directly from images of Earth scenes.

This question has been precisely adressed by Gonsalves [8, 7] who proposed a Phase-diverse phase retrieval method taking images of incoherently illuminated extended objects as input and doing a non-linear optimization of an objective function for retrieving the wavefront error. We propose, in this study, to use this method to build a real time implementation of a spaceborne active optics for Earth observation by taking an AAM aproach. Indeed, AAM is motivated by the fact that such a real-time implementation is subject to harsh space constraints (reduced onboard resources, cooling, radiations etc.) reducing the number of potential target and the operating frequency while the amount of data needed by the phase diversity algorithm is quite large and the algorithms on which the method is based (Fast Fourrier Transform, non-linear optimization ...) are quite complex.

4 Experimentations and results

4.1 Application task graph

The application task graph that we extract from the phase diversity application is described on Figure 4. The task graph is organized around five tasks and works on large images sizes (More than 200 images of sizes 1024*1024).

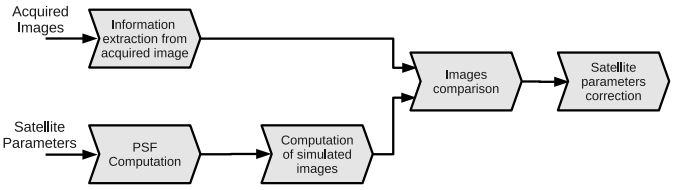

FIGURE 4 – Phase diversity application task graph

The two first tasks performed by the algorithm are to compute the PSF function based on the actual satellite parameters, and from these results to generate the simulated images.

In parallel of these two tasks the algorithm extract the relevant information from the images taken by the satellite. Which lead to the following task that make a comparison between the simulated images and the acquired images.

Based on these results, the final tasks realize an non-linear optimization to correct the satellite parameters.

4.2 Architectural constraints

Since the phase diversity application will be embedded inside an observation satellite. Constrained associated with the spatial environment must be taken into account and determine.

The main constraints are about the radiation aspects, which will reduce the number of potential target, the operating frequency, and the amount of on chip and off chip memory. Moreover being given that a satellite once set in orbit is a device that is no longer repairable, the safety under operation of the architecture is increased.

Finally still in order to reduce the weight of the satellite the power has to be minimal in order to limit the number of cooling equipment present on board. To that end the number of cores, the size of the interconnect and the amount of memory should be kept at a minimum.

4.3 Results

Based on our methodology, on the application tasks graph and on the architectural constraints, the first step of our AAM methodology allows to identify the application and architectural hotspot, which lead to the diagram presented onto Figure 5.

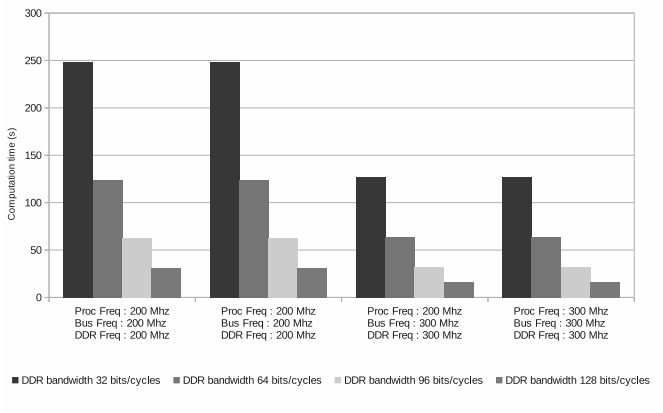

FIGURE 5 – Temporal diagram identifying the application and architectural hotspot

The results shows that the limiting factor of the performance is the external memory access.

However since in the space domain the memory is a crucial point, we then choose instead of increasing the number of external memory, to reduce the amount of data to be transferred from and to the memory.

This decision have led to a transformation of the algorithm in order to find a new operating point, and a modification of the architecture in consequence (memory size, number of cores...).

All these decisions being facilitated thanks to our methodology that provides design assistance in order to converge efficiently and quickly to an optimal solution for both the algorithm and the architecture. This have led to the results presented onto Figure 6 that met the imposed constraints.

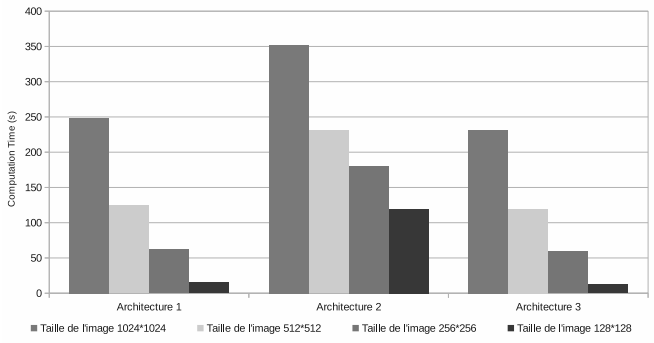

FIGURE 6 – Performance estimation obtained at the end of our methodology.

5 Conclusion and Perspectives

Through this study we observed that the DSE of heterogeneous MPSoC architecture can be particularly time consuming despite the assistance providing by existing tools.

Indeed in the context of MPSoC architectures the number of parameters that need to fixed is huge. Moreover since each architectural modification impact the algorithm specification performing the DSE by hand is almost impossible.

This is why it's necessary to introduce new heuristics and new tool that allow to efficiently construct MPSoC architecture.

Since the most difficult and time consuming task is to determine for each algorithm specification what is the best architecture and best mapping. Our proposal is to develop a tool that based on an algorithm specification and on a set of processing cores will autonomously go through the solution space and determine what is the best mapping and the best architecture for each algorithm specification and user constraints.

Then from these results, SpearDE will be triggered to generate the executable binary code, while based on a set of generic interfaces (Develop in the context of the FlexTiles project [9, 1]) the VHDL code of the architecture will be automatically generated.

Références

[1] [online]. available :http ://flextiles.eu/wordpress3/.

[2] [online]. available :http ://www.ovpworld.org/.

[3] G.Louis D. and L.Richard G. Correction of misalignment dependent aberrations of the hubble space telescope via phase retrieval. In *Current developments in optical engineering and commercial optics*, pages 201–214, 1989.

[4] J. R. Fienup. Phase-retrieval algorithms for a complicated optical systems. *Applied Optics*, 32 :1737–1746, 1993.

[5] J. R. Fienup, J. C. Marron, T. J. Schulz, and J. H. Seldin. Hubble space telescope characterized by using phase-retrieval algorithms. *Applied Optics*, 32 :1747–1767, 1993.

[6] D. Ginhac. *Adquation Algorithme Architecture - Aspects logiciels, matriels et cognitifs*. Habilitation à diriger des recherches, University of Burgundy, 2008.

[7] R. A. Gonsalves. Phase retrieval and diversity in adaptive optics. *Opt. Eng.*, 21 :829–832, 1982.

[8] R. A. Gonsalves and R. Childlaw. Wavefront sensing by phase retrieval. In *Proc. Soc. Photo-Opt. Instrum. Eng.*, volume in Applications of Digital Image Processing III, A. G. Tescher, ed., pages 32–39, 1979.

[9] F. Lemonnier, P. Millet, G. Marchesan Almeida, M. Hubner, J. Becker, S. Pillement, O. Sentieys, M. Koedam, S. Sinha, K. Goossens, C. Piguet, M. Morgan, and R. Lemaire. Towards future adaptive multiprocessor systems-on-chip : an innovative approach for flexible architectures. In *Embedded Computer Systems : Architectures, MOdeling and Simulation*, 2012.

[10] E. Lenormand and G. Edelin. An industrial perspective : a pragmatic high-end signal processing design environment at thales. In *SAMOS*, page 5257, 2003.

[11] S.Borkar. Thousand core chips : a technology perspective. *Design Automation Conference*, 978-1-59593-627-1, 2007.