Complete Heterogeneous MPSoC Synthesis

Youenn Corre*, Jean-Philippe Duguet†, Dominique Heller*, Loïc Lagadec†, Van-Trinh Hoang*

*Lab-STICC, Université de Bretagne Sud, 56100 Lorient, France
E-mail: {firstName.lastName}@univ-ubs.fr
†Lab-STICC, Université de Bretagne Occidentale, 29200 Brest, France
E-mail: {firstName.lastName}@univ-brest.fr

Abstract—The ever-growing needs of functions in embedded systems along with integration capacities lead to the development of Multi-Processor Systems-on-Chip (MPSoC). But the complexity gap of the architecture design once again raises the problem of the productivity and the lack of System Synthesis tools. In this paper, we propose a new kind of system-level design framework that semi-automatizes the design of heterogeneous Multiprocessor Systems-on-Chip. Starting from an architecture model and an application code, our framework produces a complete multiprocessor system specialized with hardware co-processors along with the executable application. One of the main innovations of our flow is the integration of High-Level Synthesis (HLS) as a step of the Design Space Exploration loop. This allows us to explore large numbers of hardware accelerators solutions and to get a fast and accurate cost estimate. We also automated other necessary design steps of the flow such as profiling of the application and the code generation for both the target architecture and the software application.

I. INTRODUCTION

GPP multiprocessor are not the ultimate architecture solution for embedded system design. For performance and energy efficiency reasons, data locality and dedicated co-processors have to be considered. However Heterogeneous Multi-Processor Systems-on-Chip raises issues that make it in practice difficult to handle. One of these issues is to build the most efficient hardware platform while respecting the constraints of an embedded system, typically area and power constraints. Another one is the efficient programming of such architectures.

Our goal is thus to make a Computer Assisted Design (CAD) tool that increases the productivity by simplifying the design of an entire system-on-chip, including architectural exploration, hardware specialization and software parallelization. To reach this goal we strongly rely on previous researches and existing tools, as pieces of the underlying design steps composing the whole flow of our framework have already been addressed in previous work. We also use as much as possible widespread and standard formalisms to ensure compatibility with other tools. After place and route, logic synthesis and behavioral synthesis, we now come to the fourth generation of CAD tools. This new generation inserts the HLS in the Design Space Exploration (DSE) loop in order to explore hardware accelerators solutions along with a fast and accurate estimate of the associated cost. By seamlessly inserting HLS in the co-design of MPSoC, along with the generation of hardware and software codes, we provide a new kind of tool flow whose inputs are a generic architecture model and the code of a software application.

II. OVERVIEW OF THE FRAMEWORK FLOW

We have defined a generic flow which is described in Figure 1. The inputs of this flow are a standard C/C++ software application code, time constraints, memory data mapping and a generic hardware architecture model to be used during design space exploration. The first step is to make an architecture independent profiling that will guide the designer in the splitting of the application into tasks, in order to express parallelism.

Then a second profiling is realized on the split code. This time the profiling is run on the application implemented on the target processor (e.g. a MicroBlaze (MB)). The objective is to extract a list of candidate tasks for a hardware implementation and to get the software execution performance values that will be used later during the co-simulation phase. The code is automatically adapted to the target processor.

The next step is the DSE loop, which is shown in figure 2. The design space is bounded by a set of specifications extracted from the provided model of architecture. So a first set of solutions for the design is generated and evaluated. A cost estimation is made, and a set of possible task mappings is generated and, using the tool Sesame [1], performances are estimated according to a defined scheduling policy. If

Fig. 1. Flow of our framework. Flow of the DSE is detailed in figure 2
objectives are not met, a new iteration is launched where a new processor or hardware component can be added to the design according to the list of promising candidate tasks. If the task corresponds to a standard function already available in the IP library then associated components with various area/time trade-offs are successively added, otherwise the HLS/DSE controller launches the HLS tool in order to generate a new coprocessor.

Finally, when both cost and performance objectives are reached, the design is delivered to the code generator which then provides a synthesizable version of the hardware architecture and the software code adapted to the architecture.

III. CONTRIBUTIONS

For the new aspects of the problem, we had to develop our own tools and techniques.

1) Profiling on Hardware: Measurements of the computing time is necessary for each task, so we realize a software monoprocessor implementation of the application on an FPGA. We implemented a code generator that automatically adapts the application to run on the target architecture. We made the profiling transparent to the application by modifying the Xilinx kernel OS to control and read a timer that counts the number of cycles taken by a task to execute. We then established a set of candidates for hardware implementation by selecting the tasks with the highest computing times.

2) DSE & HLS controller: Since we integrated HLS as a part of the DSE loop, we used a set of scripts to control and generate series of hardware accelerators by varying characteristics such as cadence, input bitwidth, etc. The HLS tool uses the C code to be accelerated as input and gives as output a RTL description such as VHDL. We also use a library of IP in order to accelerate this step even more by maximizing the reuse of components (e.g. from previous synthesis or ad-hoc designs).

3) Estimation through HLS: We also use HLS as a fast and accurate indicator for cost estimation. Using an HLS tool such as GAUT [2], we manage to compute an evaluation of the cost of an accelerator from the generated VHDL output file. So we have built and validated cost analytical models that give an estimate of resources on a given FPGA family. This model can provide a cost estimation much faster than logic synthesis and give results, which are accurate within a 10% margin. Another benefit of fast HLS-based estimation, is that it allows for trimming the exploration space as a large number of possible IP can be eliminated since only Pareto solutions are selected.

4) Code generation: Since our framework provides the executable of the application for the generated architecture, we have to adapt the application code by including API for communication and synchronization between tasks. It is also necessary to generate various file for the FPGA tools such as Xilinx’ .mhs and .mss files, which describe the hardware architecture and the mapping of the software resp. The automatic generation of code provides another important benefit: the guarantee of correctness by construction and thus that the code is bug-free.

IV. RESULTS

We validated our approach by applying our flow to an MJPEG decoder on a MicroBlaze-based architecture. After realizing the transformations on the code and getting the profiling information, we started the DSE. We first generated a series of IDCT hardware accelerators and after cost estimation, selected the only six of them that were relevant. We also generated some YUV accelerators. We then evaluated both in cost and performances several architectural solutions (1 MB with an IDCT accelerator, 2 MB with various mappings, etc.). For this example, we have explored 30 different final architectures, after pruning 64 solutions for IDCT and 4 solutions for YUV. After the designer splits the code into tasks, the whole exploration, including profiling, hardware accelerators synthesis, task scheduling and HW/SW partitioning, has required less than 10 minutes.

V. CONCLUSION & FUTURE WORK

We have presented a new HLS tool for heterogeneous MPSoC synthesis. According to a given architecture model, this tool explores and fully generates a synthesizable MPSoC architecture along with the associated software code. The main contribution is to provide a DSE loop that integrates a HLS tool as an estimator in order to explore hardware accelerators and task partitioning at the same time. We also automated tedious flow steps such as profiling and HLS control in order to reduce the design time of a complete system and thus increase the productivity.

The ongoing work mostly consists in the development of a DSE strategy that will replace the exhaustive search by a heuristic method. Other planned developments include the user interface, the introduction of new architecture models and the reuse of existing power models to be used during DSE.

REFERENCES
