Journal of Systems Architecture 60 (2014) 79–93

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

# Fast and standalone Design Space Exploration for High-Level Synthesis under resource constraints

Adrien Prost-Boucle, Olivier Muller, Frédéric Rousseau *

*TIMA Laboratory – CNRS/Grenoble-INP/UJF, 46 Avenue Félix Viallet, Grenoble, France*

ARTICLE INFO

*Article history:*
Available online 14 October 2013

*Keywords:*
High-Level Synthesis
Design Space Exploration
FPGA
Hardware accelerators
Resource constraints

ABSTRACT

The very high computing capacity available in the latest Field Programmable Gate Array (FPGA) components allows to extend their application fields, in High-Performance Computing (HPC) as well as in embedded applications. This paper presents a new methodology for Design Space Exploration (DSE) in the context of High-Level Synthesis (HLS) for HPC and embedded systems targeting FPGAs.

This new methodology provides very quickly an RTL description of the design under resources constraints. An autonomous flow is described, that performs incremental transformations of the input design description. The low complexity of the transformation evaluation, decision and exploration algorithms, associated with a greedy progression, makes this DSE methodology very fast. Moreover, this methodology respects a strict resource constraint given as bare FPGA primitive amounts. Hence, the generated design fits into the targeted FPGA or a partition of it. Such a methodology leads to autonomous, fast and transparent DSE, all these issues known to limit the use of HLS.

Results on several benchmarks highlight the capabilities of our DSE methodology. The results show a high generation time speed-up compared to one other existing HLS approach, while preserving correct performance of the generated circuits.



## 1. Introduction

FPGAs have been widely considered for integration in the field of embedded systems such as telecommunication applications, automotive, as well as High-Performance Computing [1], and supercomputers [2]. They are considered as efficient solutions, that typically provide power consumption gain and acceleration of one to three orders of magnitude over CPU or GPU implementations. FPGAs now provide huge computing capacity and require efficient design flows able to program them.

Register Transfer Level (RTL) languages are typical entry points of usual tool flows. In spite of their fitness to precisely describe all details of the architecture, RTL descriptions are cumbersome for architecture exploration. There is a huge space to give a higher abstraction description of the design removing useless details. In that case, a High-Level Synthesis (HLS) tool is required to transform this high abstraction model to RTL. HLS tools enable to reach better productivity, measured by the time to get an acceptable solution, compared to a direct RTL design. From a generic input algorithm, HLS tools usually generate an RTL architecture for a given target technology. Special user performance goals (area, frequency, throughput, power, ...) can be taken into account.

Current HLS flows enable the user to quite rapidly explore the design space [3]. As shown in Fig. 1, each RTL generation is analyzed by the user who can then command the HLS tool (with user directives) to converge towards' a better solution in the next trial. This manual user guided interaction slowdowns the Design Space Exploration (DSE) process, and is only acceptable for well-informed users. It is far from the traditional compilation process for CPU or GPU which may be done without any knowledge of the compiler or processor architecture, with an automatic tool flow. Despite their intrinsically lower performance, CPU and GPU get a better acceptance for embedded system designers due to their fast and autonomous tool flows.

Consequently, we are looking to provide a DSE methodology that is able to produce quickly an efficient RTL design, with or without user interaction.

Thus, in order to be autonomous, the flow has to strictly adapt to the amount of resources of the target. To target a full FPGA or a part of it (this latter target is mandatory in dynamic partial reconfiguration [4] or floorplanning contexts), the amount of resources combines the amounts of all FPGA primitives (e.g. LUT, FF, RAM, DSP block). Using these generic FPGA primitives enables to ease the support of new FPGA technology, obviously a desired feature of the DSE flow.

* Corresponding author. Tel.: +33 476574641.
*E-mail addresses:* adrien.prost-boucle@imag.fr (A. Prost-Boucle), olivier.muller@imag.fr (O. Muller), frederic.rousseau@imag.fr (F. Rousseau).

1383-7621/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2013.10.002

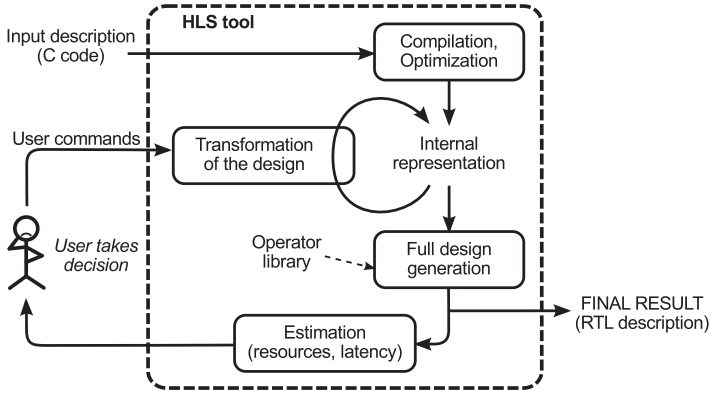


**Fig. 1.** Current HLS flows.

This paper describes a DSE methodology for HLS, which means that the targeted functionality is going to be implemented into dedicated hardware. In particular, issues involving system-level DSE or hardware/software partitioning are not tackled here, even if system-level DSE tools could greatly benefit from HLS tools featuring the proposed methodology. The proposed DSE methodology is fast and autonomous, and is able to respect user constraints (hardware resource and frequency). Conceptually, it applies to any FPGA technology, and it could be quite easily integrated in existing HLS tools.

The rest of this paper is organized as follows. The Section 2 presents the background and the related HLS tools. Section 3 describes our proposed methodology. The implementation details are given in Section 4. Performed experiments on benchmarks and results are given and discussed in Sections 5 and 6.

## 2. Background and related works

This paper deals with a rather specific topic, namely fast and standalone DSE for HLS under resource constraints. DSE involving HLS has been proposed under various forms, from the fields of HLS-DSE as well as system-level DSE. The most relevant of them are described hereafter to fully explain the choices that led to the present works.

### 2.1. System-level DSE approaches

System-level DSE approaches present interesting properties. They generally consist in deciding which functionalities of a considered system will be implemented in hardware accelerators. The targeted platforms are then heterogeneous. As described in [5], this heterogeneity leads to highly modular DSE environments, where an exploration tool specifically optimized for DSE exploits external tools as target-specific estimators (e.g. for latency and power).

A DSE environment for virtual prototyping of Systems-on-Chips (SoC) is proposed in [6]. Authors' objective is to consider functional, power and timing behaviour at system-level, under explicit partitioning and mapping to a specific implementation platform. They propose to use Matlab/Symulink with MARTE/UML models. Commercial HLS tools perform the estimations. However, no details about DSE algorithms are given and no results are presented.

Very complex DSE methodologies can be considered with exploration tools dedicated to DSE. In [7], a highly modular framework is proposed for multi-dimensional Multi-Processor SoC (MP-SoC). It can use different search algorithms to explore the various dimension of the design space, with the objective to find better solutions than with a single search algorithm.

System-level DSE approaches are certainly interesting for modularity and reusability. The specialization of these approaches for DSE can make them efficient solutions, at least for heterogeneous platforms. However, in the context of the present paper, these properties come with costs. High-level generic models lead to possible accuracy losses and to inter-tool communication overhead due to model translations. Results could also present serious sub-optimality due to low exploitation of target-specific properties. Abstraction models also makes physical platform generation difficult.

Reducing genericity can be considered to meet accuracy goals. In [8], a hardware/software partitioning DSE framework is proposed. The exploration tool is tightly coupled to an external HLS tool, GAUT [9], which generates hardware accelerators under latency or throughput constraints. For one hardware accelerator, the DSE methodology is an iterative increment of the latency constraint (in clock cycles). For each constraint value, GAUT generates a solution, and the resource usage is considered. However resource constraints are only indirectly considered, and only data-flow circuits can be handled.

Performing early estimations of the performance of heterogeneous systems is also possible. The methodology proposed in [10] facilitates the simulation of an entire HW/SW system to decide whether the effort of transforming a given computation core into hardware is needed. However their approach is focused on latency and does not guarantee that the considered application will fit into a given FPGA after actual synthesis.

The present works are focused on HLS-DSE, which means it is taken for granted that the considered application is to be implemented in hardware. In our context, the drawbacks of these system-level approaches (limited accuracy and platform generation ability, potentially high inter-tool interfacing overhead) are not affordable or inappropriate. This is why in the proposed HLS-DSE approach, exploration and estimation methods are tightly coupled and embedded in one unique tool. Besides, such a tool could fit well into a more system-level framework, as an external estimator tool for FPGA.

### 2.2. HLS and HLS-DSE approaches

As explained in [9], historically, the HLS generation flow is commonly composed of three interdependent steps: allocation, scheduling and binding. Allocation sets the number and kind of resource units in the resulting circuit (i.e. computation, storage, routing). Scheduling assigns each operation to one or several clock cycles, while ensuring data dependencies are respected. Binding