A Fast and Stand-alone HLS Methodology for Hardware Accelerator Generation Under Resource Constraints

Adrien Prost-Boucle, Olivier Muller and Frédéric Rousseau
Laboratoire TIMA - Grenoble-INP/UJF/CNRS
46 avenue Félix Viallet, 38000 Grenoble, FRANCE
email: *<firstname>.<lastname>@imag.fr*

***Abstract*—Heterogeneous architectures are common in High-Performance Computing (HPC). However, FPGA solutions are still marginal, mainly because the usual design flow are not appropriate. While preserving high speedups compared to CPU or GPGPU solutions, faster and more autonomous High-Level Synthesis (HLS) tools for FPGA are needed. In this paper, we present a hardware accelerator generation methodology that addresses these issues. It is completely autonomous and it respects area and frequency constraints. Internally, iterative transformations are applied to the design. This approach makes the Design Space Exploration (DSE) fast and scalable to complex designs. The flow has been validated with a preliminary implementation on the open-source tool UGH. The 2D IDCT algorithm has been successfully synthesized on the Xilinx XUPV5 platform.**

I. Introduction

The increasing computing capacity of Field Programmable Gate Arrays (FPGA) is opening new trends in general-purpose computing and HPC [1][2]. FPGAs exhibit a very high parallelism and allow great optimizations of the implemented algorithms. Efficient solutions typically provide power consumption gain and acceleration of one to three orders of magnitude over CPU implementations.

The recent advances in modern HLS tools gradually place HLS as the default design level, instead of RTL. For HPC, this is a big leap in productivity. Indeed, HLS tools are exploited to help Design Space Exploration (DSE) [3] and to generate the final design description with the relevant precision. Usual HLS tools generate an RTL architecture for a given target technology. Special user performance goals (area, frequency, throughput, power...) can be taken into account.

However, the current HLS flows are very iterative and the tools need manual guidance. Consequently, the DSE is still a trial-and-error process, that rests with the user knowledge and expertise. It can still take a relatively long time.

The future of HLS is much closer to the compilation task for CPUs or GPUs targets: given a hardware target (with area and frequency constraints), the HLS tool rapidly produces one unique solution. Altera has already taken this way [4]. Many applications fields (HPC, simulation, prototyping) could use FPGAs as generic hardware accelerators, but their users rarely have extra knowledge about circuit design. So a new design flow is needed, focused on automation and rapidity.

To tackle these challenges, we propose a fast, scalable and standalone methodology, able to follow precise target constraints (resources, frequency). This methodology can be integrated into existing HLS tools.

The rest of the paper is organized as follows. The section II presents the background and the related HLS works. The proposed methodology is presented in section III. Preliminary results with the IDCT design are presented in section IV.

II. Background and state of the art

Currently, most HLS tools perform their internal tasks using the following steps. First, the input design is compiled and converted into a versatile internal representation, Then, some transformations are applied according to user directives (e.g. loop unrolling [5]). Finally, the full design is generated in an RTL representation. A feedback about the quality of the design (speed and area) is provided to the designer. He can then impose other transformations to guide the tool towards a better solution.

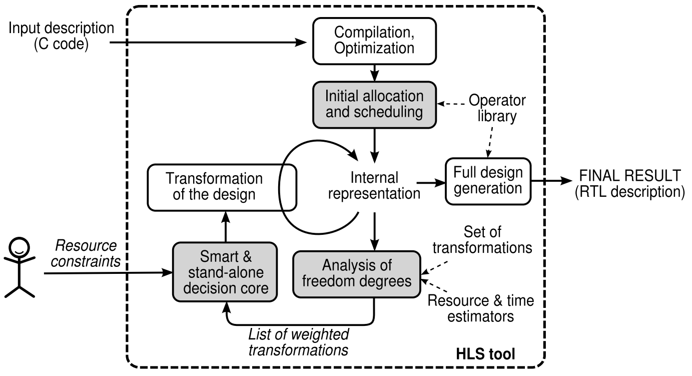
Input description (C code)
Compilation, Optimization
Initial allocation and scheduling
Operator library
Transformation of the design
Internal representation
Full design generation
FINAL RESULT (RTL description)
Resource constraints
Smart & stand-alone decision core
Analysis of freedom degrees
Set of transformations
Resource & time estimators
List of weighted transformations
HLS tool

Figure 1. Proposed implementation flow

Some of the most known tools are presented in [3]. For all of them and some others, the transformations are applied only on user command. So the DSE follows the iterative user-driven methodology described above.

Some HLS tools with automatic DSE features exist. Most of them can only handle designs with specific structures, as in [5]. DSE with evolutionary algorithms have also been proposed [6] [7], but their execution time is not deterministic, specially for complex designs. As the search tree generally grows exponentially with the design size, usual DSE techniques are not scalable.

In [8], user annotations (branch probability, loop iteration number) are used to give the tool knowledge about the actual hot spots in the design. This work is very promising as it does not require any additional user input, and the flow could reliably explore pertinent solutions. However their methodology lacks of scalability, and their flow still involves user interaction. The objectives of the present paper go beyond.

III. Overview

The proposed synthesis methodology is the combination of several features: the DSE is completely autonomous, resource and frequency constraints are strictly respected, and the flow is intended to be fast and scalable to complex designs.

For these purposes, the tool applies transformations following an iterative and greedy progression. The DSE is focused on rapidity. Only the final design generation is granted a higher optimization level.

The proposed HLS flow is illustrated in Figure 1. The main idea is the integration of the iterative decision loop inside the HLS tool in order to perform an autonomous DSE.

What is proposed is to start from a low-area circuit, obtained with a high operator sharing. Then this circuit is iteratively transformed, each time consuming parallelism opportunities and generally increasing the circuit size. The target frequency must be respected at each iteration. This elaboration process stops when the area limit is reached.

This technique allows to follow all hot spots through the entire process. Furthermore, the choices made by the tool rely on much more data than what can be exposed to the user. So potentially, an automatic process can make much more appropriate choices than a user could.

To achieve this, several new steps (colored in grey in Figure 1) are added to the usual synthesis methodology. Just after compilation, an initial synthesis step is introduced. It produces the initial low-area circuit and is meant to be fast. The internal representation is initialized with the execution time of all sections of the design, which can be inferred from the design itself or user annotations (branch probabilities, loop iterations).

Then, at each iteration of the elaboration process, the possible transformations (called freedom degrees) are detected and weighted, and the one appearing the best is applied. Each freedom degree is weighted with an estimation of the design speedup and the cost in hardware resources.

With the proposed methodology, the DSE steadily progresses towards the final solution, gradually increasing the circuit size while speeding it up. The DSE follows a greedy algorithm, which has a linear complexity in respect to the number of freedom degrees in the input design. The iterations are also very fast because the decisions and transformations are simple. These characteristics makes it possible to relatively quickly obtain a final solution even for complex designs. The obtained solution can be sub-optimal, this is the trade-off made by the proposed methodology.

IV. Preliminary implementation and results

A. Implementation based on the UGH tool

To conduct preliminary experiments, the academic and open-source tool UGH [3] is used. Its internal design representation is extended and some optimizations are performed. A characterized operator library for the target technology (here, Xilinx Virtex-5) and a preliminary elaboration core are added. The target frequency is respected with a simple retiming pass in the generated FSM. The modified tool is called AUGH, which stands for Autonomous and User-Guided High-level synthesis.

The elaboration core is able to perform three types of transformations. The first is the insertion of additional operators to the circuit. This enables to increase parallelism. The second is the wiring of conditions, which can remove some branch operations. The third type is the well-known loop unrolling.

B. Synthesis of the 2D IDCT algorithm

The 2D IDCT algorithm (original Loeffler version), with serial I/O, is used as preliminary test case. Although it is relatively small, it features freedom degrees of various types, so it is representative of the proposed methodology.

The target platform is the Xilinx XUPV5 evaluation board and the target frequency is set to 100MHz. As this circuit always uses more LUTs than Flip-Flops, the resource usage is displayed as the number of LUTs only. The embedded memory banks and DSP cores are not used here for the sake of simplicity.

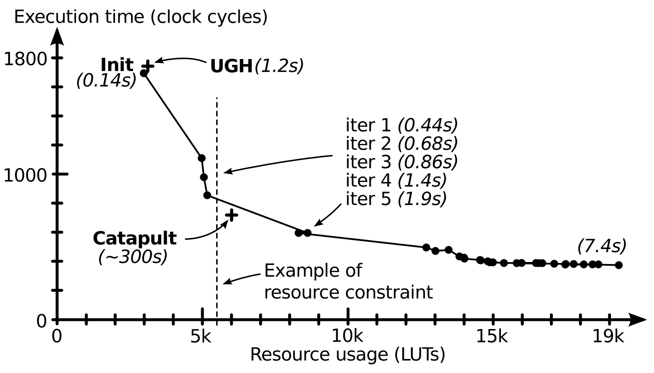

Figure 2. Elaboration steps with 2D IDCT

The Figure 2 shows the elaboration steps reached by our modified tool, when no area limit is given. It also shows the generation time to reach relevant iterations. To respect any area constraint, the DSE simply keeps the last compatible solution. This reveals the high diversity of the obtainable solutions, along with the pertinence of the proposed DSE methodology.

The result with the original UGH version is displayed for comparison, along with an experiment with the Catapult commercial tool [3]. Here Catapult is launched with no particular user command. The generation time taken by Catapult is much higher than with AUGH. Furthermore, manual trial and error experiments are necessary to perform a DSE with this commercial tool. This is highlighted with a resource constraint limit as given in Figure 2. AUGH is able to produce a solution when Catapult requires manual guidance and a much longer global generation time.

V. Conclusion

In this paper, we present a novel DSE methodology for autonomous HLS under resource constraints. The use of an iterative and greedy progression ensures a fast process. The proposed flow strictly respects user constraints about resource usage and target frequency.

A preliminary tool, called AUGH, was built. It is already able to transparently explore a wide variety of implementations and to rapidly converge towards a satisfying solution. The obtained results open up interesting perspectives to ease and accelerate generation of hardware accelerators for FPGA.

References

[1] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, “High-level synthesis for FPGAs: From prototyping to deployment,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 30, no. 4, april 2011.

[2] A. George, H. Lam, and G. Stitt, “Novo-g: At the forefront of scalable reconfigurable supercomputing,” *Computing in Science Engineering*, vol. 13, no. 1, jan.-feb. 2011.

[3] P. Coussy and A. Morawiec, *High-Level Synthesis: from Algorithm to Digital Circuit*. Springer Publishing Company, Incorporated, 2008.

[4] Altera Corporation, “Implementing FPGA design with the OpenCL standard,” White paper, nov. 2012.

[5] B. So, M. W. Hall, and P. C. Diniz, “A compiler approach to fast hardware design space exploration in FPGA-based systems,” *SIGPLAN Not.*, vol. 37, May 2002.

[6] F. Ferrandi, P. Lanzi, D. Loiacono, C. Pilato, and D. Sciuto, “A multi-objective genetic algorithm for design space exploration in high-level synthesis,” in *Symposium on VLSI, 2008. ISVLSI ’08. IEEE Computer Society Annual*, april 2008.

[7] B. Schafer, T. Takenaka, and K. Wakabayashi, “Adaptive simulated annealer for high level synthesis design space exploration,” in *VLSI Design, Automation and Test, 2009. VLSI-DAT ’09. International Symposium on*, april 2009.

[8] S. Bilavarn, G. Gogniat, J.-L. Philippe, and L. Bossuet, “Design space pruning through early estimations of area/delay tradeoffs for FPGA implementations,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 10, oct. 2006.