HLS Tools for FPGA: Faster Development with Better Performance

Alexandre Cornu\textsuperscript{1,2}, Steven Derrien\textsuperscript{2,3}, and Dominique Lavenier\textsuperscript{1,4,5}

\textsuperscript{1} IRISA, Rennes, France
\textsuperscript{2} INRIA, Rennes, France
\textsuperscript{3} Université de Rennes 1, Rennes, France
\textsuperscript{4} CNRS
\textsuperscript{5} ENS Cachan Bretagne

Abstract. Designing FPGA-based accelerators is a difficult and time-consuming task which can be softened by the emergence of new generations of High Level Synthesis Tools. This paper describes how the ImpulseC C-to-hardware compiler tool has been used to develop efficient hardware for a known genomic sequence alignment algorithms and reports HLL designs performance outperforming traditional hand written optimized HDL implementations.

Keywords: ImpulseC, High Level Synthesis Tool, Hardware Accelerator, FPGA.

1 Introduction

FPGA density is increasing exponentially in such a way that the number of gates is approximately doubling every two years. Consequently, very complex designs can consequently be integrated into a single FPGA component, which can now be considered as high computing power accelerators. However, pushing processing into such devices leads to important development time and design reliability issues. Recently, many efforts have been done to help FPGA-targeted application designers to deal with such huge amount of resources. In particular, Electronic System Level tools provide higher levels of abstraction than traditional HDL design flow. Several High Level Languages (HLL) for modeling complex systems, and corresponding High Level Synthesis (HLS) Tools to translate HLL-designs into HDL synthesizable projects are now available.

Most of them are based on a subset of C/C++ \cite{1} generally extended with specific types or I/O capabilities. This paper focuses on ImpulseC \cite{3} and its associated design flow proposed by Impulse Accelerated Technologies. It also gives feedback in the context of high performance hardware accelerator design.

Experimentations have been conducted on a specific family of algorithms coming from bioinformatics and which are known to have highly efficient parallelization on FPGA. More specifically, in this work, genomic sequence comparison algorithms are considered. As a matter of fact, many efforts have been done to parallelize these algorithms, providing many optimized implementations which can be used as references \cite{6,9,7,5}.
In this paper, we detail how ImpulseC has been used to quickly implement parallel systolic architectures. Even if here, only a single application is considered (genomic sequence comparison), the methodology we followed can obviously be extended to many other algorithms with efficient implementation on systolic array, or more generally, implementation on massively parallel architectures. To achieve reasonable performance, standard optimizations such as loop pipelining or process splitting need however to be done. The use of HLL provides an easy way to perform these optimizations, together with a fast development cycle time. However, to obtain high performance, designers have to bypass ImpulseC restrictions and perform complex program transformations.

The experiments presented in this paper show that an HLL-designed accelerator can outperform optimized HDL designs. This can be first achieved by rapidly exploring several architectural variations without great efforts compared to HDL specifications. Second, this is also achieved by the use of high level code transformations allowing the designer to generate code which better fit to HLS tool input.

The paper is organized as follows: the first section briefly describes the HLS tool we have used: ImpulseC and gives some background on the parallelization scheme used for our target genomic sequence comparison algorithm. Section 3 presents our design strategy. Performances are finally detailed in section 4 in terms of code transformation efficiency, hardware accelerator performance, and design process.

2 HLS Tool, Algorithm and Parallelization

2.1 ImpulseC

ImpulseC is a high level language based on ANSI C. It has a few restrictions, mainly on structure and pointer usage. On the other hand, it includes libraries to define constructor functions, bit-accurate data types and communication functions.

Two levels of parallelism are available: (1) coarse grain parallelism, by implementing several ImpulseC processes that can communicate through streams, signals or registers; (2) fine grain operator parallelism, within one process or one process loop, through the use of instruction pipelining and data flow parallelism.

Each process can be set as hardware process, meaning it will be hard-wired, or as software process, meaning its sequential code will be executed on a processor. Implementation of streams between hardware and software processes are managed by specific PSP (Platform Support Package). Here, two different PSP have been used: the Nios2 softcore and the XD2000i development platform.

The development environment (IDE) is called CoDeveloper. Designer can perform software simulation, generate HDL for a specific platform through the use of ImpulseC compiler, analyze ImpulseC compiler report through the Stage Master Explorer tool or generate HDL simulation testbench with the CoValidator tool.