International Journal of Software Innovation
Volume 5 • Issue 1 • January-March 2017

OpenMP-Based Approach for High Level C Loops Synthesis

Emna Kallel, CES Laboratory, ENIS School, University of Sfax, Sfax, Tunisia
Yassine Aoudni, CES Laboratory, ENIS School, University of Sfax, Sfax, Tunisia
Mohamed Abid, CES Laboratory, ENIS School, University of Sfax, Sfax, Tunisia

ABSTRACT

The complexity of embedded systems design is continuously augmented, due to the increasing quantity of components and distinct functionalities incorporated into a single system. To deal with this situation, abstraction level of projects is incessantly raised. In addition, techniques to accelerate the code production process have appeared. In this context, the automatic code generation is an interesting technique for the embedded systems project. This work presents an automatic VHDL code generation method based on the OpenMP parallel programming specification. In order to synthesize C code for loops into hardware, the authors applied the directives of OpenMP, which specifies portable implementations of shared memory parallel programs. A case study focused on the use of embedded systems for the DCT algorithm is presented in this paper to demonstrate the feasibility of the proposed approach.

KEYWORDS

C for Loops, Java Packages, OpenMP Directives, VHDL

1. INTRODUCTION

Recent trend in the modern System-On-Chip (SOC) design show that the highest performances are mostly gained by integrating multiple-processors into one die. Because there are no standard programming paradigms for SOCs (Liu, Feng, and Vipin Chaudhary, 2003), users are required to write complex assembly language for SOCs. To manage this complexity, hardware design can be performed at a higher level of abstraction. Therefore, high level synthesis tools and workbenches that accelerate and simplify the design of these new parallel systems are requisite.

Over the last decade researchers have developed methodologies, algorithms and tools aiming to generate hardware code from high level abstraction, like, parser generators, IP-Core generators, or unified modeling language (UML) tools, rapidly generating production code and decreasing design complexity (Li, Liu, Lin et al., 2015). Also, Program parallelization has become mainstream research topic due to the invention of multicore processors (Sah, & Vaidya, 2014). For example, DEFACTO (Buyukkurt, Guo, Zhi et al., 2004), SPARK (Diniz, Hall, Park et al., 2001), DWARV (OpenMP Application Program Interface, 2016) and ROCCC (Gupta, Gupta, Dutt et al., 2004) projects emphasize parallelizing transformations and some also address memory access optimizations. Also, in recent years, several commercial tools that generate hardware from HLL input also appeared (e.g. Catapult-C (Riversite optimizing compiler for configurable computing, 2016), Impulse-C (Taft, Duff,

DOI: 10.4018/IJSI.2017010101

Brukardt et al, 2007)). While these tools aim at simplifying and improving the hardware designer's tasks, their designs remains difficult and needs more expertise in this domain. So, using a standard programming model for the new parallel architecture is necessary to simplify these complex systems.

OpenMP is an industrial standard for shared memory parallel programming with growing popularity. It consists of an API enabling shared memory multiprocessing programming in C, C++ and Fortran. In other words, OpenMP API is a portable, scalable model that provides developers with a simple user-friendly interface for working with parallel applications on a wide range of platforms, from embedded systems to multi-core and shared-memory systems (B. Chapman et al, 2008). The standard API consists of a set of compiler directives to express parallelism, work sharing, and synchronization. The OpenMP provides lightweight threads and a good solution to interact with each multiprocessor computer inside various processors (Liu, Wu, Lu et al., 2015). So, it's beneficial to incorporate high-level standardization like OpenMP to improve SOC design effectiveness, and reduce the burden for parallel programmers as well.

In our research project we aim to increase the design abstraction levels by automatically generating hardware from OpenMP code to simplify and accelerate the SoC design process. In this generation process there are multiple defy to be overcome. These defy range from analyzing the high-level (HL) constructs to mapping them to the hardware. In order to lead to optimal hardware design, an efficient HL code analysis is needed. In this work, we introduce a novel JAVA-based method to parse the entered C/OpenMP code and automatically extract its different parallel constructs to be updated with the hardware components.

The next section describes the related work. Section 3 presents the proposed OpenMP parsing process. Section 4 presents the VHDL generation method. Section 5 presents experiments and results. Finally, we end up with a conclusion and future work.

2. RELATED WORK

Due to the fame of OpenMP and its powerful model allowing an easy description of parallel high-performance applications, some works have recently addressed the generation of hardware components from OpenMP programs. Indeed, a number of high level synthesis tools have been developed based on OpenMP specification. For example, the OpenMP to HandelC translator described in (Leow, Ng, & Wong, 2006) is based on a project called C-Breeze. C-Breeze, an infrastructure for building C compilers (Guyer, Jimenez, & Lin, 2001), parses a C program into an abstract syntax tree. The C-Breeze lexer and parser have been modified to accept OpenMP directives and new abstract syntax tree nodes were added to represent most OpenMP constructs. The abstract syntax tree is translated into a HandelC. The approach misses explicit memory hierarchy limiting the available memory to the resources available on chip. So that, it is hard to maintain the shared memory in an efficient and scalable manner. In this work, here are no sufficient provided details on synchronization, even as dynamic scheduling and nested parallelism. Furthermore, only the integer data type is available for use in the OpenMP description. Also, in (Burgio, Marongiu, Heller et al., 2012), HW architecture is generated from OpenMP specification, based on GCC OpenMP (GOMP) compiler (GOMP, 2016). Custom directives are proposed to specialize code regions for execution on parallel cores, accelerators, or a mix of the two. Despite the reliability of this work, there is no concrete realization like streamlining instantiation and mapping of HW tasks on the accelerated platform, such as an FPGA.

Other works cannot be directly applied to the hardware synthesis process. For example, C2SystemC tool (Dziurzanski, Bielecki, Trifunovic et al., 2006) is developed to transform an OpenMP program into a functionally equivalent SystemC description. It is based on the Cetus source-to-source translator framework (Lee, Johnson, & Eigenmann, 2003), which is a set of extensible classes particularly suited for source-to-source translators. The translator is implemented in Java language.