A Proposed Novel Description Language in Digital System Modeling

Péter Horváth
Budapest University of Technology and Economics, Hungary

Gábor Hosszú
Budapest University of Technology and Economics, Hungary

Ferenc Kovács
Budapest University of Technology and Economics, Hungary

INTRODUCTION

The increasing complexity of data-processing systems forced the design methodologies to move to a higher abstraction level (Shin, Gerstlauer, Dömer & Gajski, 2008) than the traditional register-transfer level (RTL). In the 2000s the so-called electronic system level (ESL) paradigm was evolved including the system level description languages, such as SystemC, SystemVerilog and the high-level synthesis (HLS) tools, such as SystemCrafter and Catapult-C, which are able to transform an algorithmic model into a gate level description (Casseau & Le Gal, 2012; Kim & Liu, 1995). However, at the design process of the instruction set processors the HLS method cannot be used efficiently because the main structure of the microprocessors does not fits well with the data-processing model of the digital signal processing systems, which HLS is optimized for.

Presently, the architecture description languages (ADLs) are the high-level tools of instruction set processor design. These languages can describe the functionality of the system on the behavioral level but, as opposed to the languages used in the HLS method, they can describe the structure of the system as well in the same model. Even in case of ADLs designers usually have to deal with significant restrictions in terms of microarchitecture and they are not able to model application-specific functional units with dedicated functionality.

In this article we present a novel approach of register-transfer level hardware modeling based on a new hardware description language called *Algorithmic Microarchitecture Description Language* (AMDL), which combines the advantages of HLS-based and ADL-based design approaches. It provides an algorithmic-style design entry and in the same time it makes possible to manage the exact register-transfer level structure of the designed system.

The article is organized as follows. *Section 2* gives a brief overview of concept of high-level synthesis and architecture description languages and presents the HDL representations of the register-transfer level models, which play a considerable role in the hardware synthesis methodologies. *Section 3* gives a detailed presentation of a novel hardware modeling language including its scope and objective, language constructs, model structure and design examples. Finally, *Section 4* presents a case study consisting of two application-specific instruction set processors, which were designed with the proposed design method.

BACKGROUND

High-Level Synthesis (HLS)

The HLS approach offers an algorithm level design entry and a highly automated hardware generation process in order to comply with the growing complexity and the increasingly rigorous time to market requirements. The algorithmic specification is usually described by C or C++ and the output of the design process is a synthesizable RTL hardware model. There are numerous HLS design frameworks including software tools, which are implementing these tasks, for example Cata-

DOI: 10.4018/978-1-4666-5888-2.ch686

pult C (Yuanbin, McCain, 2006), Handel-C (Abdallah & Hawkins, 2003), synASM (Sinha & Patel, 2012), SPARK (Gupta et al., 2003) and LegUp (Canis et al., 2011). The common feature of these software tools is that they speed up the design exploration and implementation phase; therefore they significantly reduce time to market and they are able to generate high quality synthesizable hardware models as well but their scope in terms of hardware functionality is limited; the HLS design method is especially efficient for digital signal processing systems (Sinha & Patel, 2012).

Although the HLS method reduces the time required to develop high quality hardware, it cannot be used in case of every type of data processing systems. There are two major problems regarding HLS: *(1) syntactic variance*, which means the synthesis result significantly depends on the coding style, and *(2) lack of interactivity*; the high-level programming language model used as design entry of the HLS, does not include detailed information about the register-transfer level functional elements and their interconnections because the exact microarchitecture takes shape only during the automated synthesis process tasks and it is difficult for designers to control this process.

The HLS method is not widely used in the design flow of general-purpose and application-specific instruction set processors (ASIPs) because in that case the foresaid problems occur even more significantly. The computation model of ASIPs does not fit well with those in digital signal processing algorithms, which HLS is optimized for (Gajski & Ramachandran, 1994). ASIPs provide a wider scale of functionality, they often contain complicated internal data-storage subsystems and their pipeline stages are more interdependent than those in the digital signal processing systems. This functional diversity increases the effect of syntactic variance. Another problem is that in case of ASIPs the optimization algorithms of the highly automated CAD tools may be not efficient enough, the time-consuming and error-prone manual optimization is unavoidable.

Architecture Description Languages (ADLs)

The main objective of the ADLs is to describe the behavior of ASIPs in order to generate software components (assembler, compiler, debugger and simulator) of the microprocessor systems automatically. Some of the ADLs are also able to describe the structure of microprocessors in a detailed form that makes possible to generate a synthesizable hardware model as well.

The nML (Fauth, Van Praete, & Freericks, 1995; Hartoog et al., 1997) provides a grammar to define the instruction set architecture (ISA) of the microprocessor and it contains information about its register transfer level structure. The abstract machine model of nML is relatively inflexible, since its implicit instruction pointer does not make possible to implement multi-word instructions and only simple pipeline can be realized with it. Another approach, EXPRESSION (Halambi et al., 1999) was originally aimed to describe processor-memory architectures but the novel versions of EXPRESSION are able to describe even a complete system on a chip (SoC) as well. The main objective of EXPRESSION-based design is to automatically develop software components for embedded systems, so it is typical compilation-oriented and simulation-oriented behavioral ADL. ArchC (Araujo et al., 2006; Rigo et al., 2004) is an open-source SystemC based ADL whose main goal is, similarly to EXPRESSION, to generate software components, primarily the ISA simulator.

Some of the ADLs are also able to underlie an automated hardware generation process. There are two major approaches in the literature for synthesizable hardware generation of ASIPs; the parameterized processor based (Sato, Hikichi, Shiomi, & Imai, 1994; Binh, Imai, Shiomi, & Hikichi, 1995), and the processor specification language based approach (Hadjiyiannis, Russo, & Devadas, 1999; Schliebusch et al., 2004). The first one defines a highly parameterized processor core template, so the freedom of the designer is confined to these parameters, such as register widths, cache memory sizes, etc. The other solution based on a mixed (behavioral and structural) description of the target architecture is more flexible, but even in this case designers usually have to deal with significant restrictions in terms of microarchitecture (e.g. data widths, pipeline structure, endianess etc.). Furthermore, these languages are optimized for machines with a stored program, therefore they are not able to model and underlie a hardware model generation of application-specific data processors with dedicated functionality.