Microelectronics Journal 46 (2015) 237–247

Contents lists available at ScienceDirect

Microelectronics Journal

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

A proposed synthesis method for Application-Specific Instruction Set Processors

Péter Horváth *, Gábor Hosszú, Ferenc Kovács

Department of Electron Devices, Budapest University of Technology and Economics, Magyar tudósok körútja 2, H-1117 Budapest, Hungary

ARTICLE INFO

Article history:
Received 13 May 2014
Received in revised form
13 December 2014
Accepted 2 January 2015
Available online 3 February 2015

Keywords:
Application-Specific Instruction Set Processor
ASIP
Architecture description language
ADL
System on Chip
SoC
RTL processor design

ABSTRACT

Due to the rapid technology advancement in integrated circuit era, the need for the high computation performance together with increasing complexity and manufacturing costs has raised the demand for high-performance configurable designs; therefore, the Application-Specific Instruction Set Processors (ASIPs) are widely used in SoC design. The automated generation of software tools for ASIPs is a commonly used technique, but the automated hardware model generation is less frequently applied in terms of final RTL implementations. Contrary to this, the final register-transfer level models are usually created, at least partly, manually. This paper presents a novel approach for automated hardware model generation for ASIPs. The new solution is based on a novel abstract ASIP model and a modeling language (Algorithmic Microarchitecture Description Language, AMDL) optimized for this architecture model. The proposed AMDL-based pre-synthesis method is based on a set of pre-defined VHDL implementation schemes, which ensure the qualities of the automatically generated register-transfer level models in terms of resource requirement and operation frequency. The design framework implementing the algorithms required by the synthesis method is also presented.

1. Introduction

Today's increasingly developing manufacturing technology makes it possible to build complete data processing systems on a single chip including digital and analog building blocks with on-chip memories and communication. These complex systems called System on Chips (SoCs) comprise various computational submodules called Application-Specific Integrated Circuits (ASICs) with a well-defined functionality, which cannot be modified after manufacturing. ASICs provide favorable energy-efficiency and high computation performance since they are optimized for a specific function. However, besides the high computation capacity, the reusability and flexibility as design constraints have gained significant importance because of the extremely high manufacturing costs.

There are two major ways to ensure flexibility in SoCs. The first approach is based on embedded reconfigurable logic devices, such as Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs), which make it possible to implement arbitrary digital functionality and can be reconfigured "on the field" after manufacturing. The other solution is the usage of general purpose devices operating according to a stored program. These devices are the well-known microprocessors whose functionality can be changed with a simple software update.

Both solutions provide flexibility on the cost of limited computation capacity. In case of general purpose microprocessors, the additional memory accesses and other administrative operations necessitated by the programmable nature cause a significant penalty in computation performance. In case of programmable logic devices the reconfigurability is achieved by generic logic cells and a high amount of programmable interconnection and wiring resources, which cause high path delays that result in a relatively low operation frequency and high power consumption.

The concept of Application-Specific Instruction Set Processors (ASIPs) is a promising result of the trade-off exploration between flexibility and computation performance [1]. ASIPs are microprocessors with a unique instruction set optimized for a specific application domain. Since they execute a program, they can quickly adapt to the varying functional requirements. At the same time they have instructions and hardware resources optimized for the target application; therefore, they provide a higher computational performance than the general purpose microprocessors [2,3].

Due to the rigorous time-to-market requirements, reducing the time-consumption of ASIP design is essential. The primary design tools of ASIPs are the Architecture Description Languages (ADLs), which are specific modeling tools for instruction sets and microarchitecture. The design frameworks based on these formal language

* Corresponding author. Tel.: +36 1 463 3072; fax: +36 1 463 2973.
E-mail address: horvathp@eet.bme.hu (P. Horváth).

http://dx.doi.org/10.1016/j.mejo.2015.01.001
0026-2692/

models mainly concentrate on the software parts of the microprocessor systems, namely the instruction set simulator, assembler, compiler, and debugger generation. The automated hardware synthesis plays a secondary role since the complex datapaths including internal data storage subsystems and interdependent pipeline stages characteristic to the instruction set processors require a high level of optimization, which cannot be achieved by contemporary Computer-Aided Design (CAD) tools. Therefore, the ADL-based design frameworks, even if they are able to generate a hardware model, often compromise or neglect the quality of the final register-transfer level (RTL) implementation [4].

This paper presents a novel approach for ASIP modeling with great emphasis on automated hardware generation. The main idea of our solution is similar to High-Level Synthesis (HLS) [5–7], where the design is described as an algorithm, then a High-Level Synthesis procedure generates an RTL model. With lowering the abstraction of the formal specification while keeping the readable algorithmic design style, our approach makes it possible to achieve a high level of optimization and a reduced development time.

The basis of the presented approach is a new abstract model of ASIP architectures and a mixed algorithm and RTL description language called Algorithmic Microarchitecture Description Language (AMDL) optimized for the proposed abstract architecture model. The algorithmic language environment of AMDL combined with the detailed description style characteristic to RTL ensures the rapid architecture implementation and the comprehensive control over the microarchitectural details as well. The initial formal language model comprises a lot of structural information, which makes it possible to generate an optimized, technology-independent RTL output, which can be transformed into a gate level model using the existing logic synthesis tools.

This paper is organized as follows: Section 2 provides a brief overview of SoC implementation solutions, ASIP modeling methodologies, and their drawbacks in terms of hardware generation. Section 3 presents a proposed novel architecture model of ASIPs and Section 4 describes a proposed synthesis approach and modeling language for this architecture model. Section 5 gives an overview of the design framework implementing the proposed synthesis method. Section 6 presents experimental results and Section 7 draws conclusions.

2. Background

2.1. System on Chip implementations

Numerous heterogeneous architectures can be created with the combination of application-specific functional units, general purpose microprocessors, and FPGA resources [8–11]. The different solutions enable a trade-off between flexibility and computation performance.

A general purpose microprocessor combined with an application-specific functional unit as an accelerator is used when certain computational tasks need a significant speed up. The accelerator may be loosely or tightly coupled to the microprocessor depending on the application. Both solutions define an interface between the two major components of the system, which may result in a bottleneck in terms of computation performance. Furthermore, if the accelerator is implemented in ASIC, both its functionality and its interface to the microprocessor are fixed.

A more flexible solution can be achieved with the application of FPGA fabric for implementing the accelerator functionality. In this case both main components provide post-fabrication flexibility but the fixed interface between the microprocessor and the programmable logic may prohibit comprehensive optimizations.

FPGA vendors usually provide another solution for combining instruction set processors with reconfigurable hardware. The soft-core processors implemented by FPGA resources provide a limited configurability in terms of instruction set, internal memories and pipeline implementation. In this case, additional accelerators can also be placed beside the microprocessor using the reconfigurable FPGA fabric. The special purpose high performance resources of the FPGAs, such as block memories, DSP slices, and high speed communication interfaces can be used either by the microprocessor or the accelerator. This solution is favorable in terms of flexibility but the interface issue mentioned above still exists and the reconfigurable nature of the hardware results in a limited operation frequency and poor energy efficiency.

2.2. Application-Specific Instruction Set Processors

ASIPs represent special types of stored-program microprocessors, whose instruction set is optimized for a certain application or application domain. This approach is similar to the processor–accelerator system but the two main parts are not separated. There is no well-defined interface between the application-specific functionality and the instruction set processor; therefore the drawback caused by their interface is completely eliminated.

There are two main approaches in ASIP design methodologies. Both of them are based on a low-level profiling of the target application. In the first case called instruction set customization, the profiling data are used to determine a subset of a general instruction-set, which the application does not use. By neglecting the unused instructions in the synthesis step the resource requirement and hence the cost and the area can be decreased. In the other approach called microarchitecture customization also known as Instruction Set Extension (ISE) the profiling data is used to determine complex functionalities the application frequently uses. Then this functionality is synthesized as a special instruction (or as a set of special instructions) implemented in highly optimized functional units called Application-Specific Custom Unit (ASCU) integrated into the processor's datapath. This solution significantly improves the computation performance.

The ASIPs incorporate the flexibility of programmable solutions and the high computational performance of ASICs. Due to the significant demand for flexibility, the rapidly developing wireless communication is one of the most important application fields of ASIPs. [12] and [13] present specific digital signal processing (DSP) architectures for decoding and demapper implementations, which can easily adapt to varying network and communication standards. [14–18] utilize the favorable computation performance of ASIPs, which can be used for efficient implementation of signal processing algorithms in multimedia applications, such as Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT), Retinex-filtering, QR Decomposition (QRD), Singular Value Decomposition (SVD) and Motion Estimation (ME). [19] presents another application field of ASIPs, namely encryption standards, which also demand a high computation performance. [20] presents a high-throughput ASIP with specialized Single Instruction Multiple Data (SIMD) instructions frequently used in biological sequence alignment algorithms. All the above mentioned works describe typical ASIP architectures in a sense that they include application-specific pipelines, which operate according to a stored program.

2.3. Algorithmic modeling of ASIPs

In SoC design industry, a widely used method for speeding up a design process is the high abstraction level design entry combined with automated design steps. In electronic system design the so-called High-Level Synthesis (HLS) [21–23] is a typical implementation of this concept. An HLS algorithm is used to transform an