

Pre-synthesis resource generation and estimation for transport-triggered architecture (TTA)-like architecture

Wei Hu^a, Yongxin Zhu^a, Zonghua Gu^{b,*}, Lei Jiang^a

^a School of Microelectronics, Shanghai Jiao Tong University, China

^b Department of Computer Science and Engineering, Hong Kong University of Science and Technology, China

Available online 21 February 2008

Abstract

Electronic system level (ESL) design is widely adopted in today's embedded systems development projects to cope with increasing system complexity and shrinking time-to-market. Even though functional verification can be performed at the system level and early design stage efficiently, it is still difficult to perform accurate hardware resource estimation. In this paper, we consider the problem of mapping an input high-level algorithm in C into hardware implementation based on the transport-triggered architecture (TTA)-like architecture, and present effective techniques for predicting architectural-level parameters and gate-level resource consumption without going through the lengthy hardware synthesis process in order to facilitate rapid design space exploration. We use some common DSP algorithms and a complete industry GPS application to show that our resource estimation results match the actual results from hardware synthesis very well, and they can be used in a feedback loop to optimize the input algorithm specification in C, e.g., the total gate count of the GPS application is reduced by 25% compared to the original input algorithm specification. In addition, the simulation results of the generated hardware descriptions in Verilog also show good agreement with the execution results of the original GPS program in C. © 2008 Elsevier B.V. All rights reserved.

Keywords: Hardware resource estimation; Transport-triggered architecture

1. Introduction

Electronic system level (ESL) design is widely adopted to cope with increasing complexity of today's embedded systems design. There are mature techniques for functional verification at the system level, but it is also very important to obtain accurate estimations of the hardware costs of the final implementation in terms of gate counts. The typical design flow starts with an input algorithm/application in a high-level language like C/C++, and generate a RTL behavior description after a lengthy process of hardware synthesis, which may take a few days even for applications of moderate size. Hardware resource information is not available until after hardware synthesis, and any change in the input algorithm requires repeating this lengthy pro-

cess, which hampers effective design space exploration. To reduce design time, the designer often makes a quick but inaccurate guess of the hardware cost of the final implementation at early stages of the design process. Accurate *pre-synthesis resource estimation* is difficult due to the huge semantic gap between the input algorithm and the final hardware implementation: the input algorithm is typically sequential, while the hardware implementation is inherently parallel. In this paper, we propose a methodology for hardware resource estimation targeting a *transport triggered architecture* (TTA)-like architecture [4]. We analyze the input algorithm in C, identify data dependencies and generate the final hardware description in Verilog. Time-consuming hardware synthesis only needs to be run once for a class of similar applications before efficient resource estimation can be performed for an application known to be similar to this class of applications.

One reason for our choice of a TTA-like architecture is its potential for low power execution [6]. In a superscalar

* Corresponding author.

E-mail addresses: huwei@ic.sjtu.edu.cn (W. Hu), zhuyongxin@sjtu.edu.cn (Y. Zhu), zgu@cse.ust.hk (Z. Gu), jianglei@ic.sjtu.edu.cn (L. Jiang).

processor, it is impossible to predict which execution pipeline will be active in the future, therefore, all pipelines must be powered on continuously. But in a TTA-like architecture, functional units are triggered by traffic tokens, so it is possible to implement a scheduler to save power consumption by power gating or clock gating the functional units based on the dataflow graph. Since power consumption is an important issue for today's embedded systems, we believe it is important to develop effective resource estimation methods for hardware architectures such as TTA. Furthermore, researchers in programming languages and compilers have also shown an increasing interest in exploiting redundancy and parallelism in the hardware architecture by providing new programming languages as well as compiler mechanisms useful for low power system design. For example, as part of the National Compiler Infrastructure Program in the US, the *Zephyr* compiler [1] supports operations on register transfer lists, which are similar to TTA operations. Another reason for choosing a TTA-like architecture is that it is fairly easy to convert a dataflow graph (DFG) to a hardware data-path on a TTA-like architecture [20], which makes hardware resource estimation for TTA-like architecture easier than for an architecture with complex and irregular datapath. For a single output, the data-path has the shape of a tree with layers of separate registers as latches and arithmetic units. If we consolidate separate registers into one register file and add bus connections, then we can obtain the data-path for a TTA-like architecture.

Fig. 1 shows our overall workflow. Given an input algorithm/application in high-level language such as C/C++, the designer first applies static analysis on the dataflow graphs (DFG) and control flow graphs (CFG) to obtain an approximated architecture, then he/she carries out design space exploration to further optimize the approximated architecture. The designer applies dynamic analysis to obtain information on the average execution count of each basic block, which is combined with static analysis results to project the runtime performance. In addition to

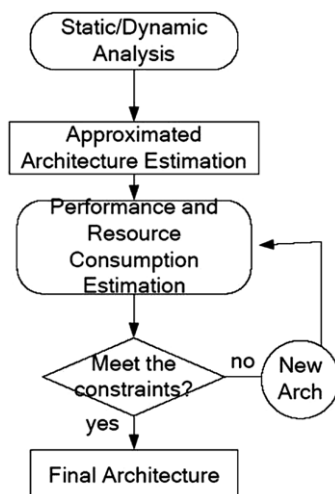


Fig. 1. Overall workflow of our methodology.

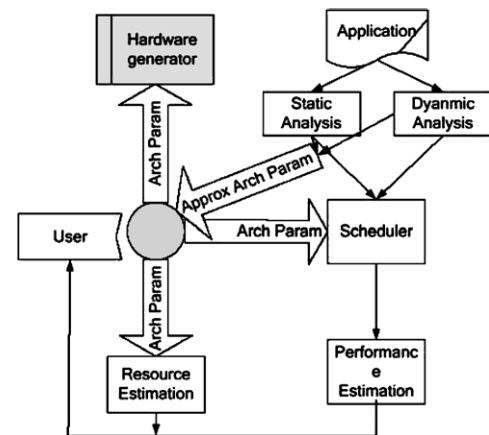


Fig. 2. Interaction between user (the designer) and our methodology.

performance projection, resource estimation can be performed to obtain gate counts for each hardware element and the whole application at the early design stage. The final architecture that meets hardware resource and performance constraints is generated at the end of the design space exploration process. In the design process, the designer only needs to search a small portion of the whole design space based on the approximated architecture parameters. Fig. 2 shows the interaction between the designer and our methodology.

The rest of the paper is structured as follows. We first discuss related work in Section 2. We then describe generation of hardware components in Section 3. We explain the resource estimation algorithm in Section 4.3, and present an industry application as a case study in Section 5. Finally, we draw conclusions in Section 6.

2. Related work

Early research on hardware generation or resource estimation [2,10] is typically based on hardware synthesis. The input algorithm is parsed to generate DFG and CFG, which are further processed through scheduling and allocation to generate control logic and data-path. Hardware resource consumption is obtained from the generated finite state machines and data-path. These early efforts are in line with the classic approach to hardware generation [20], but these solutions are only applicable to relatively small algorithms and cannot scale to systems of realistic size and complexity, since the lengthy hardware synthesis process may be an impediment to effective design space exploration.

Some later research efforts [9,11] focus on hardware–software partitioning, where hardware complexity is reduced compared to a pure hardware implementation. However, these approaches often do not explicitly take into account the operational parallelism and sharing in hardware, which may result in performance and power consumption penalties in the hardware partition. There have been some recent research on resource estimation and generation targeting new hardware platforms like reconfigurable SoCs, e.g.,

Download English Version:

<https://daneshyari.com/en/article/463216>

Download Persian Version:

<https://daneshyari.com/article/463216>

[Daneshyari.com](https://daneshyari.com)