An efficient framework for dynamic reconfiguration of instruction-set customization

Huynh Phung Huynh · Joon Edward Sim · Tulika Mitra

Received: 9 October 2008 / Accepted: 28 October 2008 / Published online: 11 November 2008
© Springer Science+Business Media, LLC 2008

Abstract We present an efficient framework for dynamic reconfiguration of application-specific custom instructions. A key component of this framework is an iterative algorithm for temporal and spatial partitioning of the loop kernels. Our algorithm maximizes the performance gain of an application while taking into consideration the dynamic reconfiguration cost. It selects the appropriate custom instructions for the loops and clubs them into one or more configurations. We model the temporal partitioning problem as a k-way graph partitioning problem. A dynamic programming based solution is used for the spatial partitioning. Comprehensive experimental results indicate that our iterative partitioning algorithm is highly scalable while producing optimal or near-optimal (99% of the optimal) performance gain.

Keywords Customizable processors · Instruction-set extensions · Dynamic reconfiguration · Temporal partitioning · Runtime reconfiguration · Custom instructions

1 Introduction

Current generation embedded systems designs are characterized by the increasing demand for higher performance under stringent time-to-market constraints. In this context, application-specific customizable processor cores strike the right balance between performance and design efforts. A customizable processor is, in general, configurable with respect
to the micro-architectural parameters. More importantly, a customizable processor may support application-specific extensions of the core instruction set. Custom instructions encapsulate the frequently occurring computation patterns in an application. They are implemented as custom functional units (CFU) in the datapath of the existing processor core. CFUs improve performance and energy consumption through parallelization and chaining of operations. Some examples of commercial customizable processors include Lx [13], ARC™ core [1], Xtensa [14] and Stretch S5 [2].

However, the total area available for the implementation of the CFUs in a processor is limited. Therefore, we may not be able to exploit the full potential of all the custom instructions in an application. This underutilization is particularly true if the application consists of a large number of kernels and each kernel requires unique custom instructions—a scenario that is quite common in high-performance embedded systems. Furthermore, it may not be possible to increase the area allocated to the CFUs due to the linear increase in the cost of the associated system. In this context, runtime reconfiguration of the CFU fabric appears quite promising. Here the set of custom instructions implemented in the fabric can change over the lifetime of the application. For multi-kernel applications, runtime reconfiguration is especially attractive, as the fabric can be tailored to implement only the custom instructions required by the active kernel(s) at any point of time. Of course, this virtualization of the CFU fabric comes at the cost of reconfiguration delay. The designer has to strike the right balance between the number of configurations and the reconfiguration cost.

To exploit this performance potential, commercial customizable processors supporting dynamic reconfiguration have been proposed. For example, Fig. 1 shows the Stretch S5 engine [28] that incorporates Tensilica Xtensa RISC processor [14] and the Stretch Instruction Set Extension Fabric (ISEF). The ISEF is software-configurable datapath based on programmable logic. It consists of a plane of arithmetic/logic elements (AU) and a plane of multiplier elements (MU) embedded and interlinked in a programmable, hierarchical routing fabric. This configurable fabric acts as a custom functional unit to the processor. It is built into the processor’s datapath, and resides alongside other traditional functional units such as the ALU and the floating point unit. The programmer defined application specific