An architecture framework for an adaptive extensible processor

Hamid Noori · Farhad Mehdipour · Kazuaki Murakami · Koji Inoue · Morteza Saheb Zamani

Published online: 1 February 2008
© Springer Science+Business Media, LLC 2008

Abstract  To improve the performance of embedded processors, an effective technique is collapsing critical computation subgraphs as application-specific instruction set extensions and executing them on custom functional units. The problem with this approach is the immense cost and the long times required to design a new processor for each application. As a solution to this issue, we propose an adaptive extensible processor in which custom instructions (CIs) are generated and added after chip-fabrication. To support this feature, custom functional units are replaced by a reconfigurable matrix of functional units (FUs). A systematic quantitative approach is used for determining the appropriate structure of the reconfigurable functional unit (RFU). We also introduce an integrated framework for generating mappable CIs on the RFU. Using this architecture, performance is improved by up to 1.33, with an average improvement of 1.16, compared to a 4-issue in-order RISC processor. By partitioning the configuration memory, detecting similar/subset CIs and merging small CIs, the size of the configuration memory is reduced by 40%.

Keywords  Reconfigurable functional unit · Extensible processor · Custom instruction · Temporal partitioning · Similarity detection · Profiling

H. Noori (✉) · K. Murakami · K. Inoue
Department of Informatics, Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan
E-mail: noori@c.csce.kyushu-u.ac.jp

F. Mehdipour
Research Institute for Information Technology, Computing and Communication Center, Kyushu University, Fukuoka, Japan

M. Saheb Zamani
IT and Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran
1 Introduction

In designing an embedded System-On-Chip (SoC), general and well-known approaches include: Application Specific Integrated Circuits (ASICs), General Purpose Processors (GPPs), Application Specific Instruction-set Processors (ASIPs), and extensible processors. Although ASICs have much higher performance and lower power consumption, they are not flexible and have an expensive and time-consuming design process. For GPPs, although the availability of tools, programmability, and ability to be rapidly deployed in embedded systems are good reasons for their common use, usually they do not offer the necessary performance. ASIPs are more flexible than ASICs and have greater potential to meet the challenging high-performance demands of embedded applications, compared to GPPs. However, the synthesis of ASIPs traditionally involved the generation of a complete instruction set architecture (ISA) for the targeted application. This full-custom solution is too expensive and has a long design turnaround time.

Another method for providing enhanced performance is application-specific instruction set extension. By creating application-specific extensions to an instruction set, the critical portions of an application’s dataflow graph (DFG) can be accelerated by mapping them to custom functional units. Custom instructions (CIs) reduce the latency of critical paths and the number of intermediate results read/written to the register file. Though not as effective as ASICs, instruction set extension improves performance and reduces energy consumption of processors due to the reduction of instruction memory accesses. Instruction set extensions also maintain a degree of system programmability, which enables them to be utilized with more flexibility. The main problem with this method is that there are significant nonrecurring engineering costs associated with its implementation. The addition of instruction set extensions to a baseline processor for each application brings along with it many of the issues associated with designing a brand new processor in the first place.

The recent emergence of configurable and extensible processors is associated with a favorable tradeoff between efficiency and flexibility, while keeping design turnaround times short. In the design of an embedded SoC, the success of product generation depends on the efficiency and flexibility for accommodating future design changes. Flexibility allows system designs to be easily modified or enhanced in response to bugs, market shifts, evolution of standards, or user requirements during the design cycle and even after production, which also means an increase in productivity. Efficiency is required to meet the tight cost, timing, and power constraints associated with embedded systems. Efficiency and flexibility are both critical, but are usually conflicting design goals in embedded system design. While efficiency is obtained through custom hardwired features, flexibility is best provided through programmable features.

The important motivation toward specialization of existing processors versus the design of complete ASIPs is to avoid the complexity of complete processor and toolset development. Xtensa from Tensilica [44], Jazz from Improv Systems [40], Stretch S5 engine from Stretch [43], ARCTangent from ARC [38], SP-5flex from 3DSP [36], LISATEk products from CoWare [39], Altera’s NIOS [37], and Xilinx MicroBlaze [46] are some commercial examples of these kinds of processors.