A Future of Customizable Processors: Are We There Yet?

Laura Pozzi
Faculty of Informatics
University of Lugano
CH-6900 Lugano, Switzerland

Pierre G. Paulin
Advanced System Technology
STMicroelectronics Inc.
Canada

Abstract

Customizable processors are being used increasingly often in SoC designs. During the past few years, they have proven to be a good way to solve the conflicting flexibility and performance requirements of embedded systems design. While their usefulness has been demonstrated in a wide range of products, a few challenges remain to be addressed: 1) Is extending a standard core template the right way to customization, or is it preferable to design a fully customized core from scratch? 2) Is the automation offered by current toolchains, in particular generation of complex instructions and their reuse, enough for what users would like to see? 3) And when we look at the future with the increasing use of multi-processor SoCs, do we see a sea of identical customized processors, or a heterogeneous mix? We comment and elaborate here on these challenges and open questions.

1. Introduction

Customizable processors have emerged in the past few years as one of the most promising components for SoC design. They have been demonstrated as an efficient solution to the stringent blend of constraints that are characteristic of embedded processor design.

So much so, that there have been articles identifying them as the one and only solution to the rise and fall of "classical" microprocessors [6]. By offering a mix of programmability and customizability, obtained by extending—or designing from scratch—simple RISC processors with custom units, customizable processors have the potential to simultaneously meet short times to market, high performance, and low power consumption. Among the best known examples of extensible processors are Tensilica [4] and ARC [2], and some levels of customizability have also been added on traditional well-established architectures (e.g., MIPS CorExtend [3] or PowerPC APUs [7]). Products have also emerged on the market to accelerate customized processor design from scratch, e.g. from CoWare[1].

2. Challenges and Open Questions

Whether the use of customizable processors will become a dominating paradigm in the near future depends on a few challenges and open questions. Some of these will be outlined here.

2.1. Customized Instructions Reuse

Even though customizable processors currently come with sophisticated toolchains, increase in automation still remains one of the important challenges. Some users are reluctant to invest precious time on manual optimizations, and request more ease of programming. As an example, the selection of custom instruction set extensions is not an easy task when left completely to the user. A few algorithms have been proposed in research in order to automatically identify extensions, making it more and more possible to select large, frequent snippets of code as highly customized instructions, at the push of a button. Commercial tools such as Tensilica's Xpress also exist for this purpose. On the one hand, the increased power of these algorithms improves custom processors performance.

On the other hand, the custom instructions that some of these algorithms can identify might be so complex that reusability becomes unlikely or very difficult: obviously, the more we customize the more we drive away from the RISC concept of simple and easily reusable Instruction Set. An important challenge is therefore that of finding reuse of complex instructions as much as possible within an application, or even across applications.

Advances are needed in instruction matching, probably not in search for exact isomorphism, but for enough similarity so that as much of the application as possible can be

978-3-9810801-2-4/DATE07 © 2007 EDAA

mapped onto complex extensions.

Of course, the problem of reuse across applications is solved, at least in theory, by implementing extensions in reconfigurable, rather than hardwired logic. If an extremely high performance reconfigurable fabric were built specifically for this purpose, it might well be the way of the future for customizable processors, as already hinted recently [6].

2.2. Design from Scratch vs. Extension of Base Core

Designing the core processor from scratch, as opposed to extending a simple, constant core, is another form of solution and comes with its own challenges. This class of cores is often referred to as Application-Specific Instruction-set Processor—or ASIP.

This approach gives a very high freedom on the overall architecture, therefore offering the advantage of a potentially higher customization level, and a tighter matching between architecture and application. Also, hardware savings can be envisioned since it is possible for designers to include the very minimum functionality required by the application.

A key challenge here is in the automatic generation of the C compiler for ASIPs. While the use of instruction extensions can be solved with in-line intrinsics, or the automated generation of these extensions, the generation of efficient C compilers for ASIPs is a more difficult problem. One solution is to restrict the ASIP exploration space to a domain-specific architecture template. This approach simplifies the automatic C compiler generation process significantly, and has been exploited effectively in commercial products, e.g., by CoWare and by Target Compiler Technologies [1, 5].

2.3. Multiple Customized Processors on Chip?

Multiprocessors systems on chips are emerging as a key class of platforms that leverage parallelism to achieve high-performance and lower power. A key debate is in the use of homogeneous vs.heterogeneous multi-processor platforms.

In the former case, a number of identical copies of a processor coexist, providing advantages such as dynamic job migration, ease of mapping, replicated design, and possibly some fault tolerance. In the latter case, heterogeneity promises better performance by proposing different cores to better match to application subsets with different requirements.

A compromise approach could be that of using multiple instances of the same customizable core family. This would provide homogeneity in the common RISC instruction-set subset shared by all processors. It also allows the use of a single development environment.

A refinement of this idea is to identify the application-specific instruction extensions that are inexpensive and duplicate them across all cores. This provides homogeneity at low-cost, especially when factoring-in the fact that memory and interconnect usually dominate the overall MP-SoC area.

Finally, more expensive instruction extensions could be duplicated on a subset of the cores, with some overprovision to allow for application performance scalability and/or fault tolerance.

3. Conclusions

Customizable processors might be the right answer to the difficult question of embedded processor design. Are we going towards a future completely dominated by customization, given the impressive advances that we have witnessed in customizable processor technology and popularity? Still, there appear to be some challenges to be met and some open questions to be solved, before this can happen. We have briefly outlined and discussed here what we believe to be the main ones.

References

[1] CoWare processor designer. *http://www.coware.com/*.

[2] T. R. Halfhill. ARC Cores encourages "plug-ins". *Microprocessor Report*, 19 June 2000.

[3] T. R. Halfhill. MIPS embraces configurable technology. *Microprocessor Report*, 3 Mar. 2003.

[4] T. R. Halfhill. Tensilica's software makes hardware. *Microprocessor Report*, 23 June 2003.

[5] Target Compiler Technologies. *http://www.retarget.com/*.

[6] N. Tredennick and B. Shimamoto. Microprocessor sunset. *Microprocessor Report*, 1 May 2004.

[7] Xilinx Virtex4 Devices. *http://www.xilinx.com*.