Embedded Computer Architectures in the MPSoC Age

Wayne Wolf
Dept. of Electrical Engineering
Princeton University
wolf@princeton.edu

1. Introduction

Embedded computers are no longer used as simple controllers. Instead, high-performance embedded processors perform complex algorithms and are linked together to form multiprocessors. Embedded computing provides students different take on computer system design because of the requirements imposed on these systems:

- Embedded computing systems generally require real-time performance. Real-time and average-time performance are very different animals.
- Battery-powered embedded systems must meet very stringent energy requirements [Aus04].
- Although the software in embedded systems can be changed to optimize the overall system, the software must also meet the specifications of the application.

As such, an architecturally-oriented embedded systems class emphasizes somewhat different concepts than a traditional, general-purpose computer architecture class. An embedded computing architecture class must use a methodology to help students quickly get their hands around an unfamiliar application. They must explore a broad range of architectures. They should also explore trade-offs between architectural modifications and software modifications to meet system goals.

Distributed embedded systems, which are built from networks of embedded processors, are also widely deployed. This paper will concentrate, however, on systems-on-chips.

1. Multiprocessor Systems-on-Chips

Multiprocessor systems-on-chips (MPSoCs) [Jer04] are, first of all, systems-on-chips. They implement complete applications on a single chip. (Although as Rich Page points out, most systems-on-chips are marketing single-chip solutions---they use one chip plus all the other chips that you need to make the SoC work.) MPSoCs are systems-on-chips that include one or more programmable processors.

Systems-on-chips are generally adapted to the application to meet performance, power, and cost goals. Although modern VLSI fabrication technology provides us with very large chips, applications keep getting larger. Some markets are large enough that specialized architectures are inevitable and desirable.

Multiprocessor systems-on-chips try to balance specialization and programmability. Programmable processors allow the SoC to be programmed after fabrication; MPSoCs are often referred to as **platforms** because they allow for many implementations of a given type of system. Programmability offers many advantages: the same chip can be used in several products, reducing product cost; design tasks can be compartmentalized; and the platform chip may have a longer shelf life than a highly specialized SoC.

Because these are systems-on-chips, they generally aren't traditional symmetric multiprocessors. They may use hardwired function units in addition to programmable processors. They may use several different instruction sets. They may have non-uniform memory spaces supported by asymmetric networks.

Many multiprocessor systems-on-chips are now available for several types of applications:

- Mobile multimedia requires both high performance and low energy consumption. The ST Nomadik and TI OMAP architectures are MPSoCs that provide specialized architectures for audio, video, and communications.
- Home multimedia is not as tightly constrained on power as mobile multimedia but requires very high performance for applications like HDTV. The Philips Nexperia architecture is a well-known MPSoC for set-top box applications.
- Networking requires very high performance and provides some opportunities for specialized parallelism. Network processors from Intel, Cisco, and others use heterogeneous architectures to process packets at high rates.

3. Architectural Challenges

Embedded computing and MPSoCs make for a full employment act for computer architects. We are in no danger of running out of applications that can make use of large amounts of computing power and that can support the design effort required to create an efficient application-specific platform. Several specific challenges flow out of our continuing need to design MPSoCs.

Configurable processors, such as those provided by Tensilica, allow the SoC designer a convenient way of quickly building processors with customized instruction sets. One area in which designers need help is figuring out which instruction set extensions should actually be implemented. Another important goal is figuring out how to connecting configurable processors into multiprocessor networks.

Hardware/software co-design [DeM01] is another way to increase system performance for a particular application. Accelerators, when properly designed, can significantly and efficiently increase performance. However, the application must be carefully analyzed to be sure that an accelerator actually improves overall performance.

Heterogeneous multiprocessors for embedded applications generally implement pipelines of processes. Our own smart camera system [Oze05] is an example of a pipelineable application. The smart camera processes video in real time, using a number of distinct steps. The amount of work performed by these stages is generally data dependent and buffers are required to smooth out rates. As video data is processed, it is boiled down in size so that data rates at the end of the process are trivial compared to the input video data rates. Pipelined application architectures bring up both hardware and software questions about buffer management and rate control.

Networks for embedded systems are another important challenge. Several networks have been proposed for on-chip use. Many of these are general-purpose networks designed to be used in many different systems. However, our own experiments indicate that asymmetric networks offer significant advantages.

Balancing generality with efficiency is a key goal in MPSoC architectures. As we pointed out elsewhere [Wol05] even relatively simple consumer devices must now implement a wide range of functions. Consider what must be performed by simple devices like digital music players or digital cameras in addition to their core functions:

- User interface.
- Cryptography.
- Networking, either through Internet or specialized protocols.
- Digital rights management.
- File systems that are compatible with PC file systems.

This wide range of functions arguably calls for a general-purpose processor; on the other hand, some of these functions may call for application-specific hardware to meet performance/power goals. We do not yet fully understand the architectural implications of the networked consumer device.

Overall, methodology is an important aspect of embedded system design that does not often come into play in general-purpose systems [Wol00]. Because embedded system designers need to design many systems and do so in a predictable amount of time with a predictable number of people, they need to develop methodologies that allow them to repeatably make reasonable decisions in new design domains. Giving students an insight into the design process can be as important as showing them specific design outcomes.

4. Benchmarks

Benchmarks are at least important in embedded computing as they are in general-purpose computing. When you are designing an application-specific system, the wrong choice of a benchmark program or input data for that program can lead to fatal misjudgments.

I believe that larger programs make more useful design examples for embedded computing for several reasons. First, high-performance embedded systems typically run several different types of algorithms; it takes a certain amount of code to exhibit all that complexity. Second, larger programs do a better job of exercising multi-tasking. Third, they give students a more realistic taste of the nature of embedded software and performance analysis.

However, it is hard to get good benchmarks and data sets. Although several reference implementations of various standards are available, they can be very hard to use. Reference implementations may make inappropriate use of dynamic memory; they may also use inefficient algorithms for critical modules. For example, many reference video encoders come with full-search motion estimation, even though that algorithm is not used in practice. Measurements made on unrealistic algorithms will lead to bad design decisions.

5. Labs

Laboratories are a critical part of an embedded systems course. As embedded systems become more

complex, it becomes harder to create an enriching set of labs for students.

Most instructors worry about the cost of lab equipment, particularly if they want to reach a broad audience. Although many microprocessor manufacturers and third parties sell evaluation boards, the associated development system is a hidden cost of these boards. Some vendors provide software along with the board while others charge a good deal of money for development systems. Ideally, students should be able to install on their own machines student versions of the development systems they use in labs; in the FPGA world, Xilinx is an excellent model for how to make devices and tools accessible to students.

Instructors can select from among a large number of uniprocessors, but it is hard to find a good experimental setup for multiprocessors. The TI OMAP processor is one of the very few embedded multiprocessors for which there exists an even moderately-priced development board, but that board is still expensive and the software environment is complex.

Much development work must be done on simulators, both in the real world and in class. Uniprocessor performance and power simulators are widely available. Although several open-source multiprocessor simulators are available, most of them are designed for symmetric multiprocessors and cannot be easily modified to handle heterogeneous multiprocessors. The MESH simulator from CMU was developed to handle heterogeneous multiprocessors as seen in systems-on-chips.

6. Conclusions

We live in an exciting time in which we have the opportunity to develop a new generation of courses on high-performance embedded computing. But because these are complex systems, instructors have to be prepared to invest time to set up lectures and labs that mate their students' interests with the applications that drive system-on-chip and large-scale distributed embedded systems. Although each institution has its own special requirements, particularly for labs, group effort may help us all build this new generation of courses.

6. References

[Aus04] Todd Austin, David Blaauw, Scott Mahlke, Trevor Mudge, Chaitali Chakrabarti, and Wayne Wolf, "Mobile Supercomputers," *IEEE Computer*, 37(5), May 2004, pp. 81-83.

[DeM01] Giovanni De Micheli, Rolf Ernst, and Wayne Wolf, eds., *Readings in Hardware/Software Co-Design*, Morgan Kaufman, 2001.

[Jer05] Ahmed A. Jerraya and Wayne Wolf, "Hardware/software interface codesign for embedded systems," *IEEE Computer*, 38(2), February 2005, pp. 63-69.

[Oze05] I. Burak Ozer, Tiehan Lu, and Wayne Wolf, "Design of a real-time gesture recognition system," *IEEE Signal Processing Magazine*, 22(3), May 2005, pp. 57-64.

[Wol00] Wayne Wolf, *Computers as Components: Principles of Embedded Computing System Design*, Morgan Kaufman, 2000.

[Wol05] Wayne Wolf, "Multimedia applications of systems-on-chips," in *Proceedings, DATE '05 Designers' Forum*, ACM Press, 2005, pp. 86-89.

.