

What and Why about Architecture for Embedded Systems

Wayne Wolf

*Department of Electrical Engineering
Princeton University*

Abstract

Embedded computing systems are becoming very common and embedded computing is starting to find its way into the curriculum. In this talk, I will discuss my views of the relationships between computer architecture education and the teaching of embedded system design. Students of embedded computing need to understand computer architecture in order to understand the characteristics of these critical components of embedded systems. The viewpoint of a user of computer architectures rather than a designer of computer architectures doesn't radically change the fundamentals that one needs to know, but it does introduce new shadings of meaning into those fundamentals.

1 Introduction

Embedded computing—the use of programmable processors in application-specific systems—has been around for quite some time. However, embedded computing courses that teach students how to design complex embedded systems are relatively new to the curriculum¹. Many modern embedded systems contain tens of thousands of lines of code; quite a few include multiple processors. Designing such systems requires quite a few skills. One of the most important skills is an understanding of computer architecture. Since CPUs are critical components of embedded computing systems, their characteristics must be very well understood in order to have any chance of creating an efficient system design.

An embedded computing course is not (at least in my view) a traditional microprocessor-based systems design course. Elsewhere², Jan Madsen and I outlined the differences between early microprocessor courses and what modern embedded system designers need to know. Early microprocessors were very limited and could perform relatively few computations; they were primarily used as I/O controllers. This introduced two biases in microprocessor courses. First, the courses were very hardware oriented and spent little time on software—usually almost no time on software above assembly language. Second, they were grounded in the details of a particular microprocessor with knowledge that could be hard to translate to a different microprocessor. Modern embedded computing courses take advantage of high-performance microprocessors that can perform very expensive calculations. This requires us to introduce software as a first-class concept. It also allows us to generalize away from the particulars of one microprocessor to the general principles underlying high-performance microprocessor systems.

Computer science and computer engineering students have always studied computer architecture. In many cases, architecture was an end unto itself. Embedded systems gives the study of computer architecture new motivation: the characteristics of CPUs help determine the characteristics of the embedded system. This doesn't require us to substantially change what we teach in architecture courses in order to help students understand embedded system design. However, the end-use application does provide a slightly different spin on the traditional subjects of architecture courses: rather than looking at problems through the lens of the CPU, embedded system designers tend to focus on the characteristics of their programs as determined by the CPU characteristics.

2 Characteristics of Embedded Systems

Embedded computing systems generally exhibit rich functionality—complex functionality is usually the reason for introducing CPUs into the design. However, they also exhibit many non-functional requirements that make the task especially challenging:

- real-time deadlines that will cause system failure if not met;
- multi-rate operation;
- in many cases, low power consumption;
- low manufacturing cost, which often means limited code size.

Workstation programmers often concentrate on functionality. They may consider the performance characteristics of a few computational kernels of their software, but rarely analyze the total application. They almost never consider power consumption and manufacturing cost. The need to juggle all these requirements makes embedded system programming very challenging and is the reason why embedded system designers need to understand computer architecture.

3 Undergraduate Education

An undergraduate course in embedded system design needs to teach students how to analyze and optimize programs for performance, power, and code size. In order to do that, students need to understand how aspects of the CPU architecture can affect these parameters. Students need a fairly sophisticated understanding of architecture in order to predict the characteristics of programs running on modern embedded processors. Those processors use pipelining, superscalar or VLIW execution, and caches to provide high performance. These features make prediction somewhat harder but not impossible.

Trace-based analysis is a powerful tool for embedded systems because it helps us identify the most time-constrained behavior of the application code. We can use traditional trace-based analysis tools to help us understand the performance and power characteristics of programs. However, analysis is only an intermediate step. After studying the analysis results, we must determine how to modify the programs to meet their requirements for speed, size, etc. These optimizations are not fully automated, so we must be able to predict where we will get the biggest payback for our effort. We must also ensure that our efforts will not have unintended consequences that will make some design parameters worse.

4 Graduate Education

Realistic embedded systems often use multiple processors: most laser printers, for example, use multiple CPUs. The basic principles of multiprocessor design are therefore an important element of embedded system design. However, embedded systems generally use custom, heterogeneous processors to reduce manufacturing cost. Because the underlying hardware architectures are not uniform, it is harder to analyze system characteristics.

Many schools now teach hardware/software partitioning courses (often as an outgrowth of their VLSI courses). These courses use FPGA boards that plug into the system bus of a PC. Custom logic can be designed for the FPGA that can communicate with the PC's CPU and memory. While this architectural template is a special case, it is a very useful domain. Students can use this architectural platform to understand the overheads incurred in communication between the CPU and FPGA, the relative performance of custom logic and programs for various applications, and the process of dividing an application into multiple processes.

Graduate education should also keep students informed about the state-of-the-art in embedded system design. Students should be exposed to state-of-the-art design. They should also learn about the algorithms used by CAD tools. Understanding the tools helps the students use them more effectively and can help them develop strategies for manual design when necessary.

Many of today's major research challenges in embedded system design come from the introduction of single-chip embedded systems. VLSI technology has advanced to the point where we can build a single chip with a complex CPU, I/O devices, and a significant amount of memory. Putting all these components makes it possible to build very high-performance systems as we eliminate pinout penalties. It also introduces new design challenges since we cannot directly measure detailed system performance by hardware monitoring. The complexity of the applications being developed for embedded computing platforms also introduces new challenges in the efficient modeling of application and system characteristics.

References

- ¹ Wayne Wolf, *Computers as Components: Principles of Embedded Computer System Design*, San Francisco: Morgan Kaufman, 2000.
- ² Wayne Wolf and Jan Madsen, "Embedded systems education for the future," *Proceedings of the IEEE*, 88(1), January 2000, pp. 23-30.