The Future of Embedded System Design

James Aylor
U. Virginia

Raul Camposano
GMD

Michael Schuette
Motorola

Wayne Wolf
Princeton

Nam Woo
AT&T

Abstract

Embedded systems use both custom software and hardware to implement product functions. Microprocessors have long been used to replace custom logic, but the range of application of microprocessors has increased greatly over the past few years—embedded CPUs range from 4-bit machines for low cost and low power to high-speed 32-bit and 64-bit microprocessors. As the scope of application of embedded CPUs changes, the design methodologies used to create embedded systems must keep pace. This panel explores what we understand of embedded system design, what problems need to be solved, and new technologies which may help solve those problems.

James Aylor

The success to the design of implementation of embedded systems lies in the development of an integrated design environment. This environment must include the capability for system-level (performance) analysis through functional analysis to implementation analysis (backward annotation). Integration implies that this environment supports true stepwise refinement and executable descriptions at each stage of the design process and enables the designer to easily develop alternatives for consideration. The modeling environment must contain a strong mathematical foundation upon which analysis through both simulation and analytical techniques can take place. It is very possible that base models may need to be transformed to other forms (Petri nets to Markov models) in order to complete the analysis process.

The successful development of an embedded systems also requires a broad knowledge of technology. Not only is it necessary to understand digital and software design, it is often necessary for the designer to know something about analog design and packaging, for example. OR have a environment that provides the designer with the knowledge and communication path with an expert. A good concurrent engineering environment that contains the support for true multidesigner development and, possibly, "expert-like" assistance tools is needed. These assistance tools can be used to aid in areas of limited expertise on the part of the designer but areas that can greatly affect the overall quality of the design.

Finally, software quality must be excellent. Software development tools must be provided that can assist the software designer in developing error-free code. It is my opinion that the software development environments do not have the level of sophistication as is found in the hardware development support tools. The level of the quality of the hardware and software developed for embedded systems should be equal.

Raul Camposano

Computers are obviously becoming a commodity. A clear symptom of this fact is the crises that the computer industry is currently going through: high profit margins are a thing of the past and revenue is not raising as fast as in the past. This is good news for most of the industry. Embedded systems in particular become more attractive with falling hardware costs. Therefore, I expect a variety of new applications. Embedded systems add tremendous value to systems (where money can still be made). Examples are the automotive sector, tool machinery and computer based control (airplanes, power plants, trains, etc.).

Key to the design of embedded systems is the integration of specific application knowledge (engineering), hardware (computer, digital and analog electronics) and software. Future CAD solutions will hopefully take into account all these aspects. Flexible frameworks that allow an easy integration of tools in the above fields will play a central role. I envision a good solution as an environment that allows to integrate easily software development tools, hardware CAD and DA, and domain specific tools (e.g. mechanical CAD). An simulation environment that also provides a "virtual reality" to validate designs in a simulated environment is one of the ultimate goals.

We are just beginning system CAD. Integration, i.e. flexible and broad frameworks that allow the integration of virtually any tool, is just beginning to be investigated seriously. Visualization of results (if I design a TV I want to see the TV picture, if I design an engine control I want to...
see the engine operating, etc.) in a wide range of applications and in real time is far from being practical yet. CAD and DA have also a long way to go - particularly to allow a very fast design. Of all the above I think integration will be the key issue.

We need vast amounts of computing power. Software development and frameworks must evolve considerably to allow us to use it in a scenario as depicted above. Simulation is established as a basic approach in science (as theory and experimentation), but there is little general knowledge that allows to build simulation environments that include many domains in a rapid, efficient manner.

Embedded systems are becoming one of the economic motors of the future. Designing them rapidly and efficiently will become an increasing need for any industrialized society. I expect a strong development of system design CAD in the future (as hardware CAD in the last 20 years).

Michael Schuette

The embedded system of tomorrow will more than likely be a portable computing platform which is in communication with other computing platforms through a radio link. The computing platform can range from a pager to a laptop computer. Such types of embedded systems add new dimensions to the system design constraints: real-time timing, power consumption, weight, EMI, size, etc.

Existing software engineering, synthesis, and analysis technologies can help in the design of such a system but must be modified to support design of the system along these added dimensions. As a prerequisite to this, significant work needs to be done in the formulation of models that describe the system in these dimensions and that can be manipulated by the design tools. Finally, closer integration of all tools must be achieved so that the end product correctly implements the system specification and changes in the product are directly reflected in changes to the specification.

Wayne Wolf

I think embedded system design today resembles VLSI design of fifteen to twenty years ago. When Mead and Conway started preaching the VLSI gospel, each chip design was seen as a separate project. Designers were gurus who kept everything in their heads and tended to translate the design from architectural conception directly to layout. In contrast, we now tend to categorize chip design in terms of architectural style (gate array, standard cell, macro cell). We have design methodologies for these various styles which carry us through all the steps necessary to create a manufacturable chip.

Today, embedded system designers tend to take a one-off approach to their designs; it is difficult to generalize the results of one project to give lessons on how to build another. I'd like to see a more top-down, analysis-driven approach to embedded system design. Today, the best we can do in many cases is to write a bunch of code, build the hardware platform, and start debugging; if we guess wrong on a design decision, we often don't find out until the initial implementation is complete. I would like to be able to build a high-level model of my embedded system, analyze its performance and cost, and refine that model through successive stages to an implementation.

In short, what we need is software CAD—CAD tools which help us design embedded software. Modern programming tools are targeted at large systems where compilation speed is paramount and implementation costs (processor size, memory, etc.) are not tightly controlled. Embedded software more closely resembles integrated circuit design; the implementation must meet strict memory size constraints; performance improvements come at substantial cost, and the system may have to meet hard real-time constraints; tradeoffs between features, performance, and size are critical; and designers are more willing to wait while CAD tools optimize the design. Embedded systems have many challenges which make them substantially different from VLSI designs, but I believe the CAD metaphor will help us drastically improve the productivity of embedded system designers in the 1990s.

Nam Woo

Embedded systems designed at AT&T range from home telephones controlled by 8-bit microprocessors to high-speed switching systems. No one set of tools will satisfy all these requirements, but some common techniques probably underlie all such designs. Our experience suggests that most designs can be divided into three parts: functions clearly destined for hardware, functions clearly destined for software, and co-design functions which may end up in either hardware or software. Current design practice does not give system architects many tools to explore hardware-software partitioning. As a result, they use intuition or rules-of-thumb to decide whether each co-design operation is to be implemented by hardware or software. Trade-offs that can be exploited during the system partitioning is usually ignored and, as a result, system designers usually do not get the best implementation.
We want to develop tools which help system architects explore the co-design space. A good partitioning between software and hardware implementations depends on many factors including performance, cost, maintainability, flexibility, size, etc. It is desirable to have an automated system that provides a variety of partitions, each satisfying the above constraints in different degrees. Then, system architects can choose the best solution that meets their requirements.