

Virtual Systems Prototyping Ensures Reusable Design Platforms

By Linda Prowse-Fosler

Abstract

This paper describes the traditional, sequential development environment and its shortcomings with respect to true concurrent hardware and software development and verification; outlines the concurrent development process and the use of virtual systems prototyping; describes a virtual systems prototype in depth; and shows how a virtual systems prototyping development process helps to ensure the business goals of maximizing time to market through platform-based design. The paper concludes with a look at several examples of projects that have benefited from the virtual systems prototyping development process.

The effectiveness of platform-based design

When asked about the effectiveness of platform-based design, a world-wise and battle scarred engineering director said, "It's just great...once you have a platform worth reusing." When asked about the value of prototyping, this cynic added, "Given the amount a [hardware] prototype costs and where it comes in the schedule, I'd be better off just shipping the hardware."

As the world of complex systems design continues to grapple with painful tradeoffs -- the cost of development, functionality, usability, quality, time to market, robustness, power consumption and a host of other factors -- the quiet revolutions of platform-based design (PBD) and virtual systems prototyping have been advancing from the fringe of early adopters to the mainstream of design. These two techniques make a big difference individually and are even more powerful used together.

In its simplest definition PBD can be thought of as the canonization of the informal architectural process used for decades by creative-but-too-busy hardware, software and mechanical engineers to create a new control system for a product. For years these folks have begun new designs by reaching into their desks and pulling out the last, similar, "good" design to see how much could be reused.

Jean-Marc Chateau of ST Microelectronics has defined platform-based design as "...the creation of a stable microprocessor-based architecture that can be rapidly extended, customized for a range of applications, and delivered to customers for quick deployment." The

concept of a hardware platform-based design has improved time to market and reduced costs at ST, Nokia, Erickson, TI and many other organizations.

However, now that software dominates in the development of most complex electronic systems, the notion of a platform must be expanded also to incorporate software: drivers, operating systems, communication stacks, and middleware. In fact, software reuse is actually more urgent than hardware reuse, for software development is where most of the product development costs reside. What developers need is a "stable hardware and software architecture that can be rapidly extended, customized for a range of applications..." to paraphrase Jean-Marc. Virtual systems prototyping enables this.

A virtual systems prototype is a software-simulation-based, timing-accurate, electronic systems level (ESL) model of the electronic system, used first at the architectural level and then as an executable golden reference model throughout the design cycle. Virtual systems prototyping enables developers to accurately and efficiently make the painful tradeoffs between that quarrelling family of design siblings functionality, flexibility, performance, power consumption, quality, ergonomics, schedule and cost.

The real cost of traditional, sequential, development

The nine-step diagram in Figure 1 shows a conventional, sequential, electronic system development process. Some argue that this process really has seven steps, or possible 11, but few would disagree that however many steps you envision, four things are true:

First, software architecture, design and implementation are last and late in the process.

Second, the design work done at the systems architecture level is redone for hardware and then reinterpreted again for software. (This potentially needless rework also has the undesirable attribute of risking the introduction of error in each, usually manual, translation).

Third, you can't conduct definitive systems level test until the software is completed.

Fourth and last, until the whole system is ready to ship in volume, no one makes any money.

And even these four truths do not represent the whole story. The traditional development process is so inherently error prone that teams rush through the early phases of architectural design and exploration to “save” enough time for the lengthy debug cycle. In many organizations, only 5% of the total development effort is devoted to system level architectural design and exploration and a total of only 15 % of the rest of the product life cycle is typically spent in hardware and software design and implementation. The rest of the effort and schedule is consumed by the test, debug and reimplement cycle. Despite this focus on testing and debugging, less than 10% of complex electronic systems actually ship on their originally targeted completion date...and most have at least one respin of the chip.

Being able to efficiently achieve full architectural exploration and design as well as true concurrent hardware and software development and verification is the key to a new level of platform reuse.

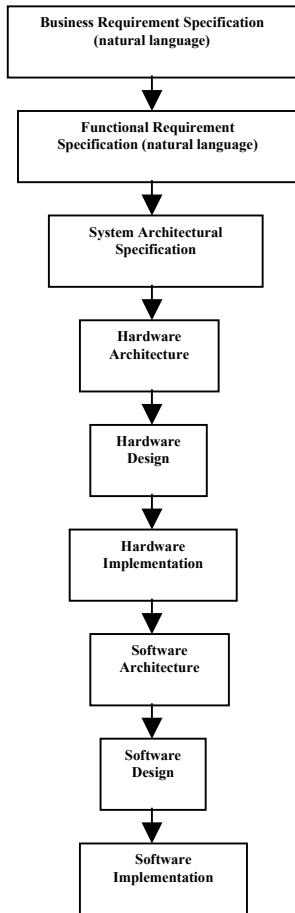


Figure 1

Concurrent development and virtual system prototyping

A concurrent development life cycle is shown in Figure 2. The first two steps, business and functional requirements, usually written in natural language, are the same as the conventional model described previously. Where this process begins to diverge is at the “Initial Systems Architecture” step where a model is built that becomes the executable system specification. This executable system specification is then used to map the system functions into hardware and software as appropriate.

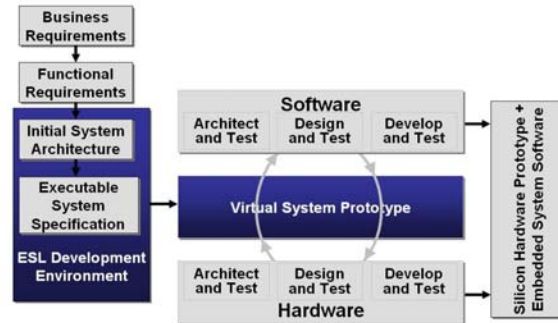


Figure 2

The architectural model/executable specification becomes both the golden reference design to drive the hardware development and the prototype on which embedded software is developed. It is a virtual system prototype that facilitates decisions at all levels: architectural, the design stage, and both the software and hardware implemented at the detailed, structural level.

The virtual system prototype (VSP)—rather than a hardware prototype—acts in place of the final system hardware until final integration, where real silicon hardware is used. Because the VSP has been used throughout the design cycle, final integration is both relatively easy and short. In this essentially top-down process, test generation from the system level to the detailed design level occurs at each stage as part of the development process. Automating this top-down process has the additional benefit of speeding up design and producing good documentation for maximum reuse potential.

A deeper look at virtual systems prototyping

A VSP is a software-simulation-based, timing-accurate design level model of part of a system or an entire system that can be used concurrently by many, if not all, of the engineering design team members. They can develop hardware, software, test software, test equipment, and verification and validation vectors. As depicted in Figure 3, concurrent development in the absence of physical hardware enables an earlier start for the many engineering disciplines involved in the product design process and results in a shorter overall design cycle.

The VSP can be used early in the development process to better understand hardware and software partitioning decisions and determine throughput considerations associated with implementations. Early use of functional models to determine microprocessor hardware configurations and architectures, and the architecture of ASIC in development, can aid in capturing requirements, improving functional performance and expectations. Working through application and system issues at this

stage of silicon development easily can save a silicon revision, or “respin,” prior to production. Depending on the technology, this can save \$50,000 to \$1,000,000 in mask charges alone.

The VSP also promotes the efficiency of a global engineering workforce through its ability to be almost simultaneously ported to or accessed by engineering resources located in facilities worldwide. Finally, the VSP provides a new level of product visibility that is not typically available to all members of the engineering team. Since the entire product is created in a simulation environment, access to internal probe points within the design is almost unlimited.

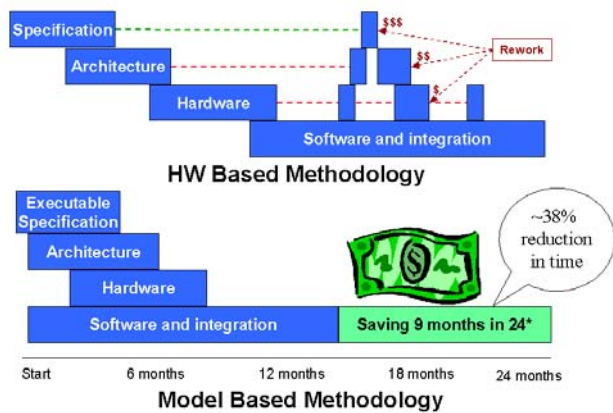


Figure 3

Virtual systems prototyping for wireless device development

The wireless device market is one area where the reduction in time to market provided by VSP has been especially useful. In cellular phone development, a conventional, serial, hardware-based methodology requires a semiconductor supplier typically to spend one year designing, fabricating and developing basic software for a microprocessor. When the device is ready, it is sent to the handset manufacturer who integrates the device and further develops the system software. Several turns in silicon are required during this process and it can take another year for the system to be fully integrated and tested. Figure 4 shows a detailed view of this process.

Figure 5 shows the detailed development process that results from using a virtual system prototype methodology. VSP methodology enables the semiconductor supplier to deliver a virtual prototype of the device to the overall system integrator nine months earlier in the process. This means that the system integrator can start software development nine months earlier and typically can complete the final validation at

the same time the hardware is ready. At least one turn in silicon can be eliminated with this process.

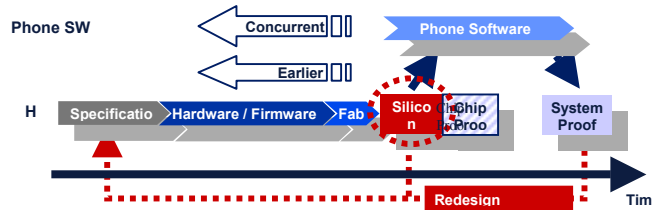


Figure 4

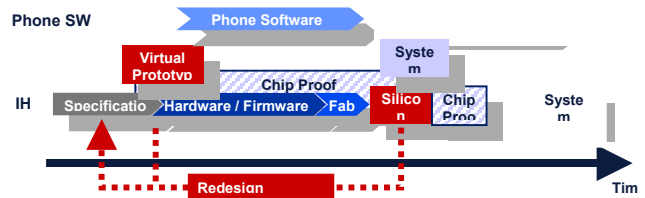


Figure 5

Conclusion

In this paper we have explored the elements of virtual systems prototyping and reviewed how a virtual systems prototyping development process expands the definition of platform-based design by making it possible to include the drivers, operating systems, communication stacks, and even middleware. Such expansion is critical since as the software component of complex electronic systems is expected to grow more rapidly than the hardware component for most future systems. Further, we have shown how a virtual system prototyping methodology and tools can improve quality all along the development process through more complete architectural design and analysis, improved hardware and software design, earlier verification and substantially improved development team communication. A wireless application demonstrates how virtual systems prototyping can cut development schedules and reduce cost.

The trend of the 1980s and 1990s towards ever-smaller geometries enabling ever more complex designs continues. The methods and tools used to create the wireless phone of yesterday, or even today, will not be sufficient for the instrument of tomorrow (and the same applies to cars, cameras, robotized vacuum cleaners and Susie’s next Christmas toy). By designing more thoroughly at the architectural level, and using a golden reference throughout the development cycle, a virtual system prototype becomes the platform for preserving and enriching the initial design and future designs throughout the product life cycle.

1. Frank J. Winters, Carsten Mielenz, Graham Hellestrand (2004): Design Process Changes Enabling Rapid Development, SAE Convergence 2004 proceedings, Convergence Transportation Electronics Association, October 2004
2. Graham Hellestrand,(2004): How Virtual Prototypes Aid SoC Design, EE Design, May 2004
3. Gabe Moretti (2003): Platform-based design: Blocks and buses lead the way, EDN, August 2003
4. Jean-Marc Chateau (2001): Flexible Platform-Based Design, EE Design, February 2001
5. Alberto Sangiovanni-Vincentelli, Grant Martin (2001): Platform-Based Design and Software Design for Embedded Systems, IEEE Design & Test of Computers, Nov./Dec. 2001
6. Alberto Sangiovanni Vincentelli, The Edgar L. and Harold H. Buttner Chair of EECS: (2000), Platform-based Design, University of California at Berkeley
7. Graham Hellestrand, (1999): The Revolution in Systems Engineering, IEEE Spectrum, September 1999