



The System Engineering Inflection Point

Virtual-Platform-Driven Concurrent Engineering

January 2004

Introduction

The increasing use of virtual platforms and the declining use of hardware prototypes define an inflection point that characterizes a paradigm shift in embedded systems design. High-performance, cycle-accurate virtual platforms, together with tools that enable a paradigm shift in the engineering of embedded electronic systems and silicon products have been shown to:

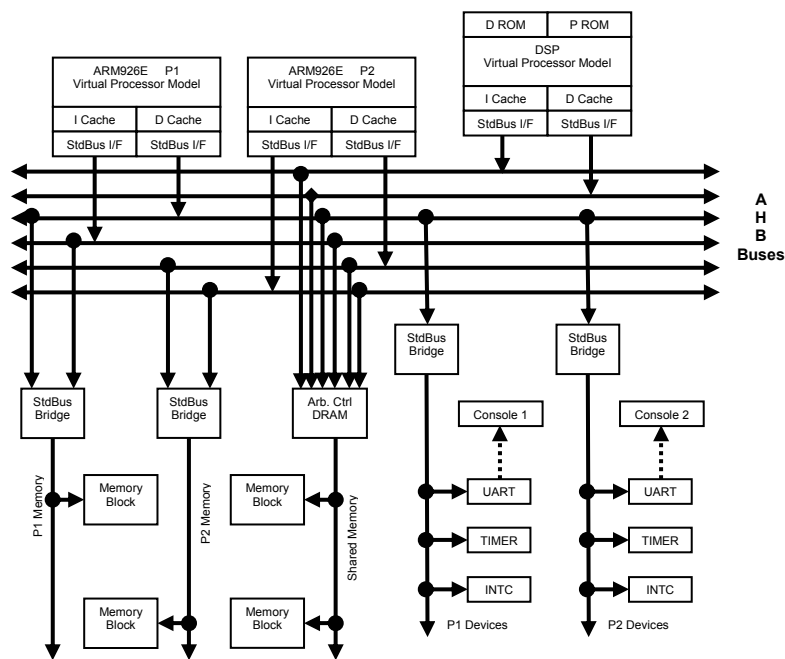
- Reduce product development risk by about 67%
- Reduce development bench cost by about 50%
- Reduce engineering development time by about 25%
- Reduce engineering resources by about 25%
- Improve product quality by 50%

Most electronic systems engineers have a hardware-centric view, resulting in hardware-centric design methodologies. But a broader system view is more appropriate. More electronic systems are incorporating complex subsystems containing large amounts of control software. This effect is driving a greater need for efficiencies in the engineering processes required to deliver high-quality electronic subsystems combining hardware and software into a reliable, high performance control system. This paper makes the case for adopting a paradigm shift in the engineering process that is underpinned by using virtual platforms instead of hardware prototypes for embedded software development.

Virtual Platforms: What Are They?

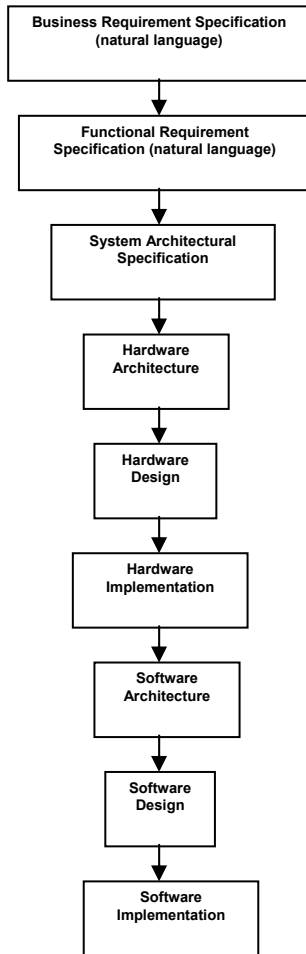
A virtual platform is a software-simulation-based, architectural-level model of the electronic system. The diagram on the right describes the architecture of a typical modern virtual platform. The model can include processor(s), hardware peripheral components, and even models of mechanical subsystems that are part of the overall system.

The virtual platform runs the same compiled and linked target code as does the real hardware. In addition, the virtual platform is cycle-accurate so that the system under design is modeled for real-time requirements.



A Typical 3G Wireless Virtual Platform

Traditional Sequential Engineering Process



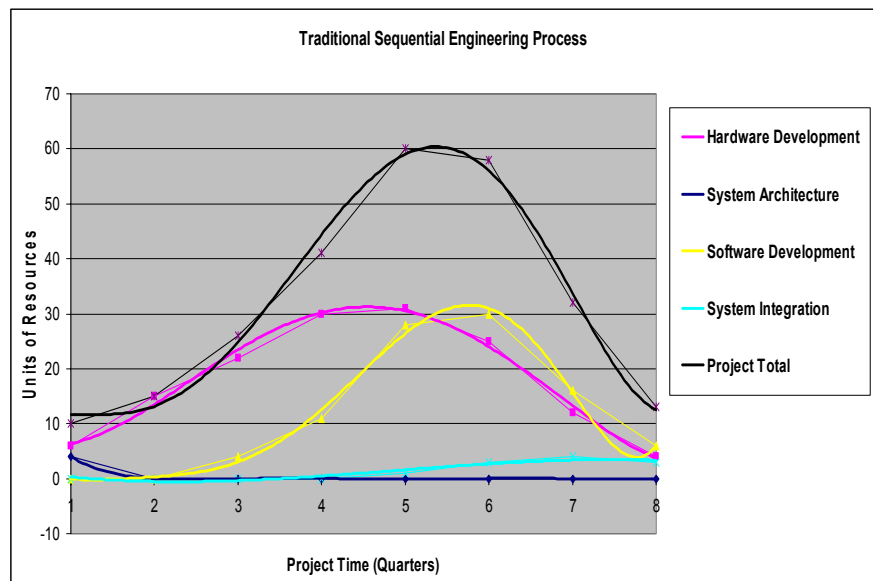
Traditional Sequential Engineering Process

The conventional electronic system development process is shown in the diagram to the left. The process begins with both business and functional requirements written in natural language. Once this stage is complete, the overall hardware and software architecture is determined—typically manually. This is followed by the hardware and software development, both of which contain the standard process elements—architecture, algorithmic design, and implementation.

The curves to the right below reflect the typical resource usage and schedule for products built using the conventional engineering process. In the diagram, the engineering process has been normalized to cover eight quarters. The resource units represent fully-loaded personnel plus development equipment and tools.

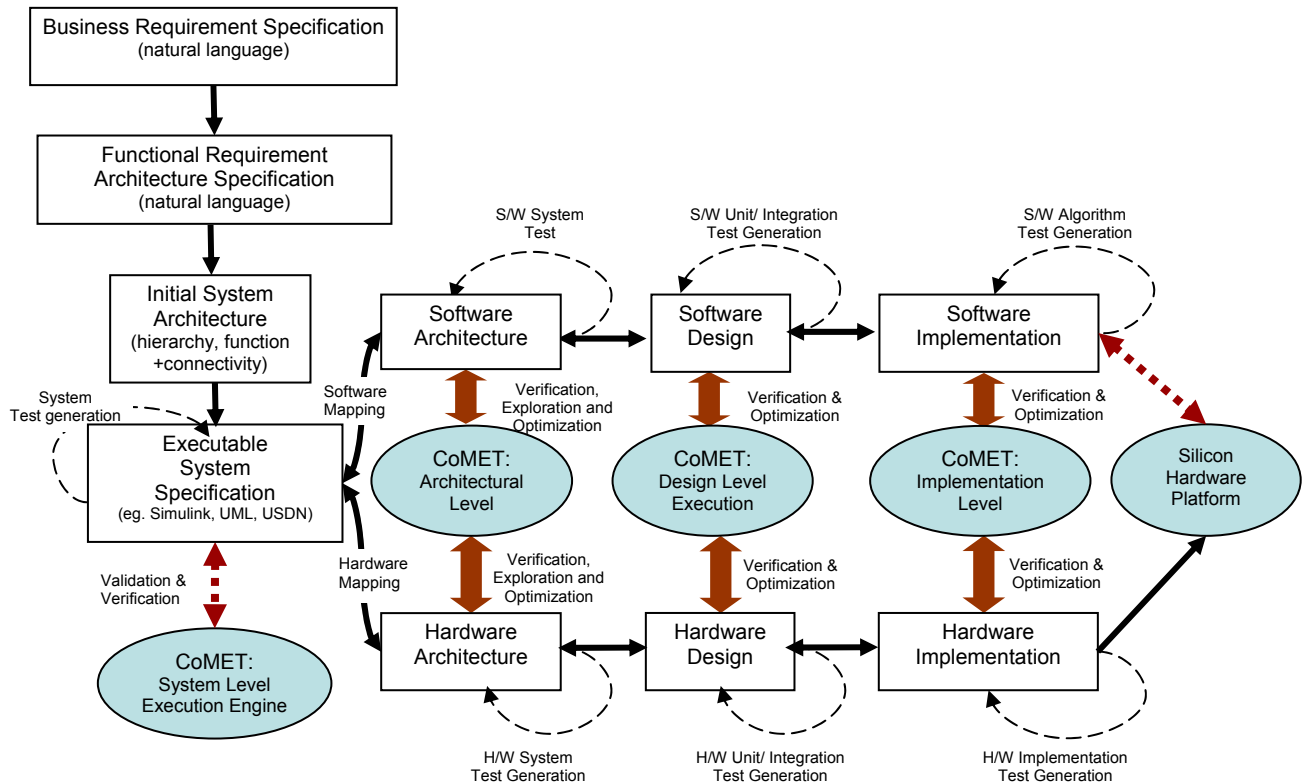
Of particular note in the conventional process: very little time and resources are applied in architecture development. Instead, the bulk of the early effort is focused on hardware design. Software development starts in earnest at the beginning of the fourth quarter and peaks at the beginning of the sixth quarter, well after the peak in hardware development. This is because software development requires the existence of a hardware prototype as a vehicle for software creation and debugging.

The area under the individual curves represents the resources used in each of the engineering activities. Notice that the total resource curve peaks some time in the fifth quarter, which is rather late in the overall process. The risk associated with a project is a function of the standard deviation of the total resource curve and the time at which the peak of this curve occurs.



Paradigm Shift: Concurrent Engineering Process

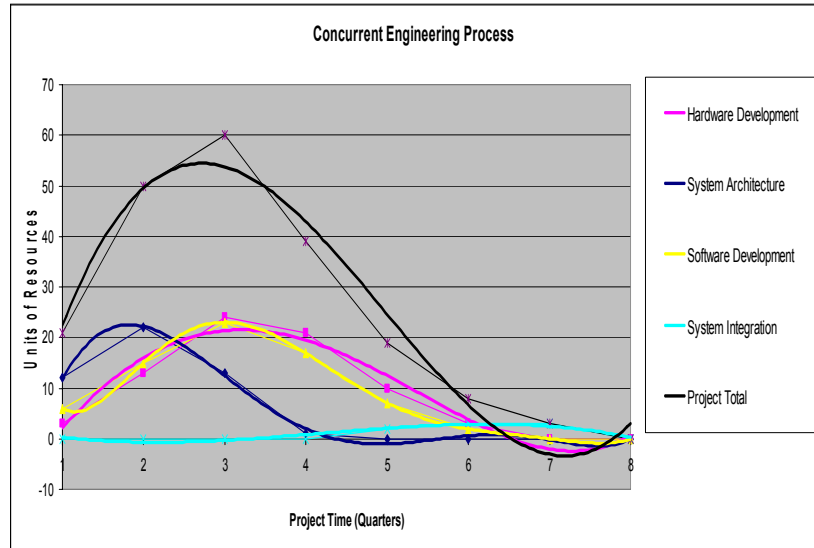
The concurrent development process is shown in the diagram below. As with the conventional engineering process, it begins with both business and functional requirements written in natural language. An initial system architectural model is built and this model becomes the executable system specification. The various system functions are mapped into hardware and software as appropriate.



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. The architectural stage of development includes exploration of various architectures on both the software and hardware sides. Once the architectural decisions are made, the development process moves on to the design stage. Finally, both the software and hardware are implemented at the detailed, structural level.

During all three design phases, the virtual platform—rather than a hardware prototype—acts in place of the final system hardware. Only in the final integration stage is the real silicon hardware used. Since both the software and hardware development are completed using a virtual platform, the final integration stage is both 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.

The resource curves for the concurrent engineering process are shown to the right. Separate resource curves are shown for architecture, hardware design, software development and system integration, along with a total resource curve. Notice that in the concurrent development process, a much higher investment is made in architecture. This principally entails the creation of a software-simulation-based architectural model of the system.



Once the model is created, both hardware and software development can ramp up simultaneously, use fewer overall resources, and complete sooner. Furthermore, it should be noted that the peak in total resources occurs late in the second quarter—about 2-1/2 quarters earlier than in the conventional system engineering process. The concurrent engineering process is heavily dependent on the development of fast, accurate system-level models and the CoMET and METeor development environments from VaST.

Overwhelming Advantages of Adopting the Paradigm Shift

Engineering Productivity

Accurate, high-performance virtual platforms not only enable the overlapping of software development with hardware design, but also deliver a unique new element to system-level design—the capability to rapidly model and quantitatively assess various candidate architectures leading to the choice of an optimal architecture without having to build the physical system. This ability to determine the optimal architecture is the linchpin to providing both a more leveraged engineering process and more economically competitive products.

The paradigm shift is characterized by four new dimensions of increased engineering productivity:

- 1) Minimization of the risk of making a major investment in a poor design.
- 2) Elimination of the overengineering of products arising in the traditional sequential engineering process from the lack of quantitative design data at the start of the design process.
- 3) Elimination of the wasteful engineering work arising in the traditional sequential engineering process due to the lack of a common communication medium. The virtual platform provides a common, executable specification for the engineering of the system and its constituent hardware and software.
- 4) Ability to project future engineering trends onto the virtual platform and thereby specify a family of future designs.

Return on Investment

Naturally, the switch to a virtual platform-driven concurrent engineering methodology requires some initial investment that must eventually return dividends. The principal investment is in the resources required to create and quantitatively assess system-level architectural models running full application software in order to determine the optimal system design. The payoff for making this investment is directly observable by comparing the respective curves for the two development processes. The most noteworthy features are the following:

- The investment in up-front architectural modeling is larger in the concurrent process. This investment enables a go/no-go decision point early in the development cycle.
- Hardware and software development occur essentially simultaneously. This greatly enhances the efficiency of designing software with hardware.
- The total resource curve is dramatically shifted earlier in the concurrent process, which directly reduces the risk associated with the project.
- The greater efficiency and reduced risk indicated by the curves in the concurrent process result in fewer requirements for resources and earlier product development completion.

This is summarized in the table below:

Parameter	Net Effect of Concurrent Process
Overall Risk	Reduced by ~67%: The computed risk is proportional to the standard deviation of the total resource curve and inversely proportional to the square of the fraction of project time remaining to complete the project, where time is indicated by the peak of the total resource curve.
Development Bench Cost	Reduced by ~50%: The cost of typical bench tools such as prototype boards, logic analyzers, in-circuit emulators and distribution of prototypes is halved.
Development Time	Reduced by ~25%: Overlapping hardware and software design allows for early completion of the development cycle.
Resources Required	Reduced by ~25%: "Overdesign" is eliminated because reliable, quantitative performance data is available early in the process. Design efficiency achieved by overlapping hardware and software development results directly from lockstep communication of hardware and software engineers, reducing rework. Software engineers are more productive because software prototypes are more easily stimulated and more observable than hardware prototypes.
Product Quality	Improved by ~50%: Since the architectural model is, essentially, an "executable specification," confusion over specification is eliminated. Significantly increased testing and validation is enabled by the availability of a fast system model very early in the development process.

Conclusion

The advent of high-performance, cycle-accurate, virtual platforms is permanently changing the landscape of embedded systems development. The increasing use of virtual platforms and the declining use of hardware prototypes define an inflection point that characterizes a paradigm shift in embedded systems design. VaST is the technology leader in virtual platforms, providing the highest performance virtual platforms and system engineering tools to the embedded systems design community.