

The Engineering of Supersystems

Graham Hellestrand, Vast Systems Technology Corp.

The average car today contains from 20 to 80 computers executing a couple hundred million to several billion instructions per second. Typically, these computer systems communicate through two to four networks that use various protocols with varying packet transit times, bus bandwidths, and failure tolerances.

Even an everyday mobile phone contains two to four processors executing several hundred million instructions per second (MIPS) in closely coupled or networked configurations that implement the mobile modem as software on a single digital signal processor (DSP).

The base stations controlling wireless and wireline communications systems are themselves a hierarchy of closely coupled systems with multiprocessor—typically DSP—subsystems executing billions of instructions per second. A complete base station can incorporate from five to 20 subsystems and 100 separate processors.

DESIGN CRISIS

These *supersystems*, incorporating possibly dozens of processors in closely coupled or networked topologies, pose a design challenge at least equal to that of designing the component systems and processors. Supersystem design and verification must address hardware complexity that increases with each successive generation of a product family, as well as embedded software content that increases exponentially with time.



A sequential hardware-software process fails for embedded designs in which software development dominates the engineering.

Embedded software implements the real-time control functions that support product safety-critical and failure-tolerant requirements, among others, and defines the critical path for bringing many supersystems to market. Thus, for the design and verification of embedded systems generally and supersystems specifically, the availability of a hardware prototype late in the sequential design process—usually after silicon design is complete—renders the traditional engineering process high risk and unreliable.

A recent report showed that more than 50 percent of embedded system developments run late and 20 percent either fail to meet their requirements specifications or are cancelled (J. Krasner, “Embedded Software Development Issues and Challenges,” Embedded Market Forecasters, 2003; www.embeddedforecast.com/emf_esdi&c.pdf). These percentages point to a crisis in embedded system design and a failure in the underlying engineering process.

In conventional system development, hardware design precedes software development. This sequential process is failing for systems in which

software development dominates engineering. The sequential nexus between hardware design and software development institutes a straight-jacketed process that does not scale to the engineering of supersystems.

A development methodology that addresses the requirements for these systems must solve three problems:

- hardware becoming available late in an engineering process that has

software development on the critical path;

- hardware exhibiting increasing generational complexity; and
- several-thousand-page documents written in a natural language specifying a complex product, such as cell phones, with a total design cycle of less than one year.

These problems are exacerbated by competitive pressures that narrow market windows and by consumer demand that reduces time to market for many embedded system products.

EXECUTABLE SPECIFICATIONS

The past 10 years have seen important advances in system-level modeling. We can now create high-performance, timing-accurate, complete system models that do more than just mimic hardware architecture—they can actually define the architecture as a *virtual system prototype*.

The VSP is derived from a product's business and functional requirements, as Figure 1 shows. Using system-level design tools, system architects can produce accurate facsimiles of the hardware and software—a mixture of

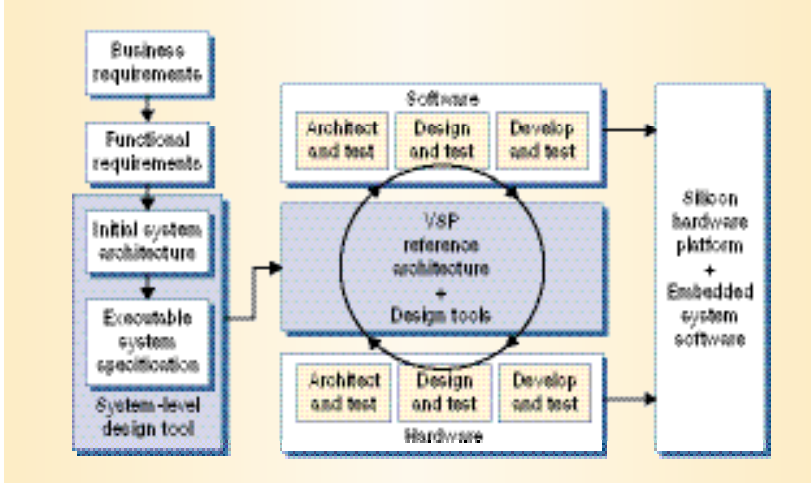


Figure 1. VSP-driven system engineering process. A virtual system prototype drives concurrent, iterative hardware and software development.

actual code (the operating system, for instance) and skeleton code (applications and middleware databases). This executable system specification, or VSP, can then serve as a golden reference model, driving concurrent hardware-software development and supporting hardware and software optimization for complex control tasks.

Creating an optimal VSP requires a quantitative methodology capable of feeding candidate VSPs into test regimes that run extensive software workloads and measure the results. System architects analyze the results and incrementally improve the next iteration by, among other changes, systematically varying architectural parameters that partly characterize the VSP.

This process determines an optimal hardware-software architecture, or family of architectures, destined to just meet a new product's market requirements. The process relies on VSPs that are functionally complete, inherently timing accurate, and capable of software execution performance greater than 10 MIPS. It can be partially automated by using the experimental design statistics to drive, for instance, the selection of the next VSP parameter set.

The VSP executable specification leverages the engineering process when used to concurrently drive the

hardware design and software development processes. With the concurrent development processes, evolving software can execute on all hardware models developed during hardware architecture and detailed design. Conversely, evolving hardware can be tested using any software developed during software architecture and detailed design. This turns the conventional development and verification/validation process on its head: The systems-level test cases are generated as part of the VSP specification, then the architectural, module, and unit tests are generated from the top down, as hardware and software development progresses through the engineering process.

VSP VERSUS CONVENTIONAL PROCESS

To compare the VSP-driven process with the conventional sequential process, consider two curve sets depicting the resources deployed for developing a multiprocessor 2.5G mobile phone. Typically, about 30 percent of the engineering effort for such developments is hardware design and 70 percent is software development.

In Figure 2, each graph contains separate resource deployment curves for each process activity:

- architecture specification and assessment,
- hardware design,
- software/firmware development,
- system integration (with verification and validation), and
- overall project (integration of activities).

The conventional process is back-end loaded, as the overall project curve in Figure 2a shows. The architecture is subsumed by the front-end hardware design process, and software ramps up in the second half of the process after hardware design.

This sequencing of processes is inefficient, since software invariably uncovers shortcomings in the architecture and hardware design. System rework costs increase exponentially as a function of how late in a project problems are uncovered, so the risk is high in the conventional process.

By contrast, the VSP-driven process is front-end loaded. Specification and architecture constitute the initial major effort, and both precede hardware design and software development, which proceed in parallel.

Table 1 compares the process metrics of note for the project. The risk factor is computed as the square root of a function of two parameters: the overall process volatility, or standard deviation, and the inverse square of the fraction of the project remaining to be completed at the time of maximum resource deployment.

In a successful VSP-driven process, all processor and bus models must be timing accurate, and processor models must run at between 30 and 200 MIPS on an off-the-shelf home PC.

EXAMPLE SUPERSYSTEM

Vast Systems Technology designed a networked VSP to demonstrate the application of VSPs in architecting and engineering distributed automotive control systems.

The VSP features six processor-centric, timing-accurate automotive *electronic control units*. ECUs manage

activities like air-bag deployment and antilock braking; they are connected by dual controller-area network (CAN) buses. Each ECU can run any software suitably compiled and configured for that controller and can send messages to other ECUs via either CAN bus.

Such VSPs support the development of optimal architectures for particular car subsystems by quantifying trade-offs among performance, cost, quality, safety, distribution of functionality, tolerance of ECU or CAN bus failure, and engineering process risk. Other important uses include developing time-critical software for distributed control algorithms, designing the hardware that realizes the hardware platform, and verifying that communications bandwidths and processing capabilities are sufficient to meet the ECU's real-time deadline requirements.

This particular VSP executes software in each ECU at about 5 MIPS (about 30+ MIPS overall processing capability). The ECUs run software, including operating systems such as Linux, iTron, OSEK, and Nucleus; real-time control algorithms; and test communication code for a range of workloads.

Metering these complex supersystem VSPs yields invaluable information that is often impossible to gather from a physical system, including bus contention and bandwidth utilization; working set match/mismatch to cache size and set associativity; latency in system response to internal and external events; correlation of external, hardware, and software conditions and events; and transaction and algorithm throughput.

Supersystem designers use such metrics to drive the iterative VSP improvement process that precedes committing the design to silicon. The ability to analyze errors and anomalous behaviors during long regression runs in the first 20 percent of a supersystem's development, rather than in the last 20 percent, avoids the prohibitive costs of rearchitecting, redesigning, and reimplementing products—especially in 90-nanometer silicon.

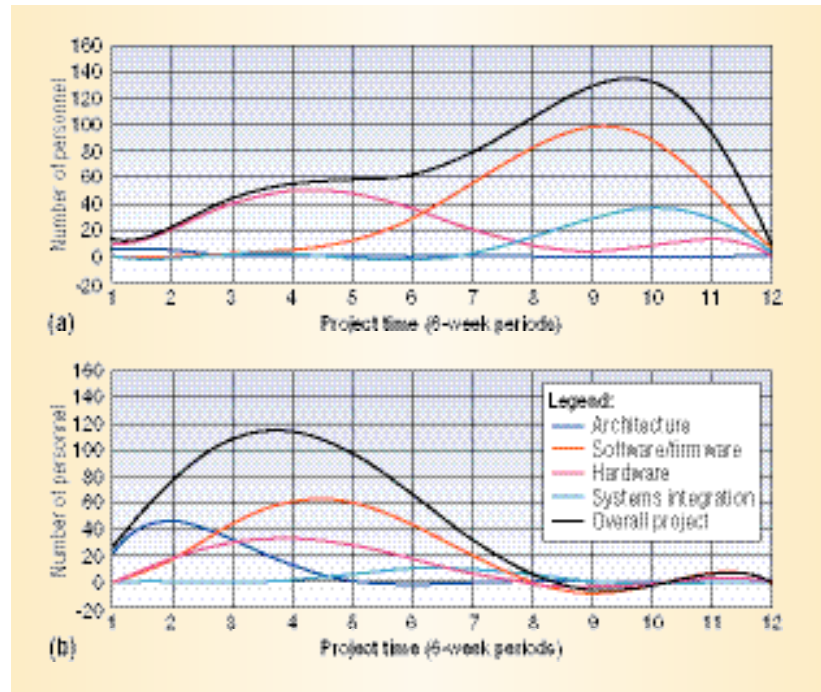


Figure 2. System development process comparisons: (a) the conventional design process requires most resources toward the end, and (b) the VSP-driven process is front loaded.

Table 1. Comparison of resource usage between conventional and VSP-driven development processes for the 2.5G mobile phone project.

Metric	Conventional process	VSP-driven process
Total resources deployed	800 person weeks	520 person weeks
Peak resources deployed	135 engineers at month 14	115 engineers at month 4.5
Risk factor	40	10
Total development time	18 months	13 months

The VSP engineering process achieves significant productivity gains and a factor-of-four mitigation in project risk. It also supports the competitive imperatives of building efficient and cost-effective hardware-software systems for real-time critical control applications—as well as for any software development that must precede or be coterminous with hardware design.

VSP can revolutionize systems engineering by providing high-accuracy, high-performance virtual prototypes early in the engineering process. It

is the sine qua non technology and methodology for engineering supersystems. ■

Graham Hellestrand is the founder and Chairman of Vast Systems Technology Corporation. Contact him at g.hellestrand@vastsystems.com.

Editor: Wayne Wolf, Dept. of Electrical Engineering, Princeton University, Princeton NJ; wolf@princeton.edu