

# The Quantitative Design Process in Embedded System and Strategic Engineering

Graham R. Hellestrand  
VaST Systems Technology Corporation  
1250 Oakmead Pkwy, Ste 310, Sunnyvale, CA 94070 USA

## Abstract

The engineering of embedded systems optimized for particular purposes involves the mastery of multiple engineering disciplines and technologies, including: software engineering, digital and analog electronics engineering, networked systems engineering, operating system and compiler technology, processor micro-architecture technology, multi-processor platform technology, simulation technology and (i) for wireless products – radio frequency technology and engineering, (ii) for automotive control products – mechanical engineering and complex distributed control systems.

The methodologies and tools for building optimal systems has begun to be available over the past 3-4 years and, where deployed, are having a revolutionary impact on companies and industries. As the effects of economic globalization become more pervasive, the differentiation of successful companies will focus more on the excellence of engineering and its ability to rapidly build purpose driven families of software-hardware electronic controllers optimized for families of products within and across market sectors.

This paper deals with the technologies, methodologies and tools required to enable engineering teams to develop a competitive, often *first-mover*, advantage in the highly competitive global market. The use of engineering as a competitive advantage – read weapon – will distinguish the winners from the losers through this protracted market inflection point.

## 1. Embedded System Engineering

**Definition:** *Embedded systems* are control systems embedded in products and devices that are implemented using software and hardware. Apart from network connectivity and local memory, embedded systems are usually stand-alone and have prescribed interfaces for user interaction and programming.

An empirical approach to composing optimal architectures for application specific embedded systems is relatively rare. The use of empiricism in developing optimal software is even rarer, and when used often primitive. The complexity of processor centric, electronic systems that control modern products (such as, cell phones, automobiles, communication base stations, consumer products) requires a systematic approach to developing the system (both hardware and software) in order to deliver an optimal fit for an intended product. When a company's engineering process is being used as a competitive weapon, the luxury of optimality, especially wrt power and speed and response latency in mobile systems, rapidly becomes a necessity [1].

The bigger architecture picture is more complex. The intuitive optimization of systems – architecture, software design, hardware design, and interfaces – has largely been a by-product of hardware design. Since hardware designers have rarely understood, or had access to, the software that would run on their architectures, they produced conservative, often grossly over-engineered designs that were typically poor fits to a number of dimensions of the specification - especially in cost sensitive applications, where over-engineering is the antithesis of cost sensitivity [2].

The ability to support data-driven decision making early in the systems development process is one of the underlying drivers of building models of systems that are timing accurate and high performance. Optimizing systems across the dimension of speed, response latency, cost (size) and power consumption is rarely done and, at a pre-silicon level, it is an undertaking only possible using high-performance, timing accurate models – called Virtual System Prototypes (VSPs) in this paper.

It is known that poor software and inefficient algorithms have a 1<sup>st</sup> order effect on an embedded system's performance, as does hardware architecture (bus skeleton, memory hierarchy, etc.) at the system level. This is difficult to reconcile with practice, when next-generation product planning often has prime focus of processor microarchitecture, regardless of the fact that iterative microarchitecture improvement typically yields a 2<sup>nd</sup> or 3<sup>rd</sup> order effect. The question is what has happened to software in optimizing systems?

Embedded systems may be very complex as the dual processor network switching subsystem in Figure 1, below. Even though the hardware may be complex in such modern systems, the software is yet more complex and dominates the engineering process and budget. It is negligent not to employ empirical methods as a normal part of the development of software in real-time, embedded systems.

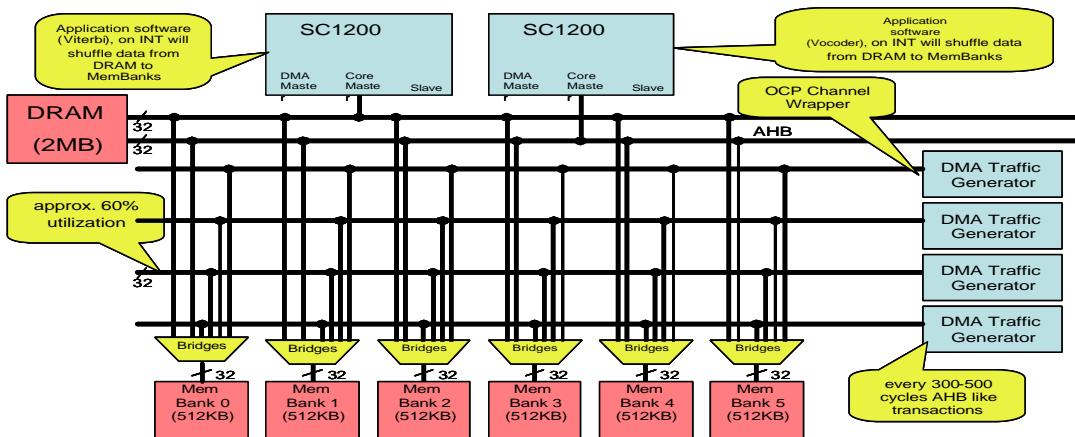


Figure 1. Network Switched Subsystem

## 2. The Systems Design Process

The design process of systems that control complex products needs to embody the development of an executable specification of the system, early in the process, so that empirical evaluation of candidate architectures can be undertaken prior to committing the controller to a physical realization. The architecture produced by this process becomes the golden reference design and the process requires compliance by all designs derived from the specification, including register transfer (Verilog and VHDL) designs. The other primary impact of this process is the complete overlapping of hardware and software development and the consequent reduction in time-to-market (TTM) that the parallel development enables. This engineering process, shown in Figure 2 below, has shown unusually positive impacts when implemented, especially for products that have ever shortening design cycles.

Some features of the design process are worth noting. The process starts conventionally with Functional Requirements being derived from Business Requirements. Next an Initial System Architecture is created that complies with the

Functional Requirements. This is an abstract, often mathematical model of the system that is executable and empirically testable. Such models are constituted from inter-communicating tasks/processes created using Simulink, UML, or some other programming system. The 1<sup>st</sup> version of the Initial Executable Architecture is likely to have sufficient characteristics to be able to represent the underlying system with (i) a high level of functional fidelity and rudimentary timing or (ii) with a high level of timing fidelity and rudimentary function. It is imperative that executable models created in the Architecture Driven process are finely instrumented so that measurement of timing, behaviour and underlying event activity is readily available from all levels of the model. This data is used to drive the iterative refinement of the abstract system - resulting in high levels of both function and timing fidelity. What the abstract models lack, intentionally, is underpinning physical realization detail, more of which becomes defined during the process of mapping the Initial System Architecture to an Executable System Specification or Virtual System Prototype.

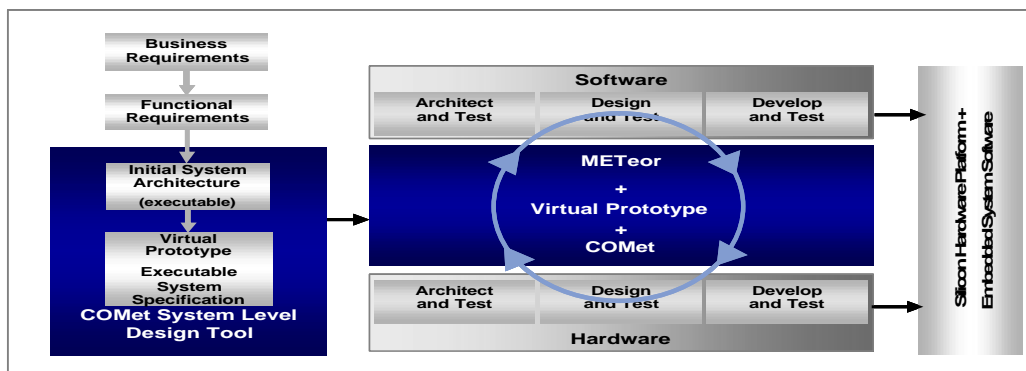


Figure 2: Architecture Driven Design Process

The NP-hard problems of mapping an Initial System Architecture to underlying *physically-mappable* structures, realized as software and models of electronic hardware, mechanical, radio-frequency, and other devices (the VSP), is a current area of intense research activity [3]. All of the credible mapping strategies involve empirical experimentation using actual or stub software and models of the electronic hardware and other devices. The driving of the structural, functional and timing parameters of the software and hardware subsystems using *Design of Experiments* [4] techniques leads to *optimized* systems for specified purposes. It is important for the VSP to be *optimized* prior to commitment to physical realization as software + silicon or FPGA + external devices. Attempts at post-facto system optimization during realization are expensive and dangerous and leads to significant degradation of the engineering process.

The surviving VSP is now the *golden reference model* to be used concurrently for software development and iterative refinement of the electronic hardware (called a Virtual Prototype or Platform) to an implementation in silicon or some other physical technology. The inherent concurrency in the engineering process reduces the effect of software being almost always on the critical path in modern electronic systems engineering.

A comparison of the Conventional Hardware → Software Design Process and the architecturally driven Quantitative Systems Engineering Process, with quantitative empirical experimentation driving *optimization*, is shown in Figure 3, below. The immediate impact is the movement of peak resource deployment in a project from the last third of a project to the first third. This is due, mainly, to the up-front development

of an *optimized* system architecture that enables a running start in software development, including porting of legacy code and operating systems, and an early start on reducing the Virtual Prototype hardware to physical reality. The data used in deriving the Conventional Process curves is from the development of a 2.5G cell phone. This data is reused to normalize data from a customer building a 2.5G cell phone, but using the Quantitative Systems Engineering Process. The comparison is not perfect, but the strong difference is verified in practice.

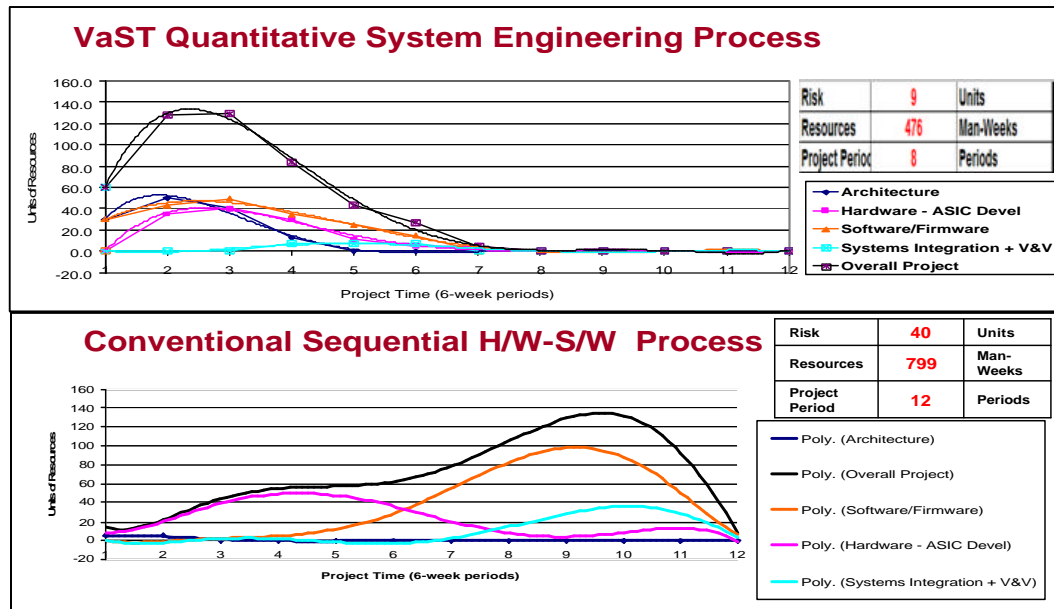


Figure 3. Comparison of Conventional and Architecture Driven Design processes

A comparison of the summary statistics for each process is revealing:

Measure	Conventional Process	Quantitative Systems Engineering Process	% Improvement
Risk	40	9	78
Resources	799	476	40
Project Duration	18 months	12 months	33

Risk is computed as: the square root of the (variance of Overall Project Resource curve divided by the square of the fraction of project remaining at the point of peak resource deployment). The use of variance, as a conservative measure, parallels the use of the concept used in the Black-Scholes option pricing model in which high option prices reflect the risk associated with their purchase.

The mitigation of risk is the biggest factor. And this, together with the 33% reduction in time-to-market makes an overwhelming case for the adoption of the Quantitative Systems Engineering Process.

### 3. Strategic Engineering – Purpose Driven Optimization

Strategic Engineering is defined as: The ability to optimize products through structuring families of controller architectures using quantitative processes (typically empirical) and the utilization of innovation to minimize time-to-market, resources required, and project risk.

The optimization of systems at the Virtual Prototype level, requires the formulation of a specific objective function (such as, maximize speed, maximize throughput, minimize power). The scientific method is about rejecting hypotheses using a rational, data driven decision making process. One of the challenges in making decisions in this engineering domain is the complexity of modern super systems and identifying patterns in, and making sense of, the potentially billions of pieces of data collected from hundreds of unique sources of measurement of platform activity and latency, available from the silicon and simulation.

On the optimization side, there are many ways to construct objective functions. The classical way is to track event frequencies and/or latencies and to construct ad hoc functions based on functionally related events, such as CPU events, bus and bus-bridge events, memory events, device events, etc. A more systematic way is to use the multivariate statistics to help formulate dependence relations based on abstract concepts and more concrete factors derived from the interpretation of highly correlated events measured during simulation or silicon activity. The latter approach is beyond the scope of this paper.

#### 4. Measuring Systems – The Enabling Metre

The fundamental premise of optimization is that there is an ability to accurately measure the system being optimized. In a Virtual System Prototype, the requirement is to be able to probe processor models, bus and bus bridge models, device models, software algorithms and structure. These are the elements that enable decisions to be based on data.

Measurements are typically of functionality (such as cache activity) and time (such as intervals between two significant events). Sequences of measures are also important to provide history-based optimization computations.

An example of a history-based optimization function appears below in Equation 1. In an event driven simulation environment, a general form of an optimization function (in this case power) can be expressed as a function whose parameters are functions each characterizing contributions to the objective function by one of the components constituting the system, viz. CPUs, buses, bus bridges, memories and peripheral devices. This equation is based on functions ( $f_x$ ) that take occurrences of event sequences that correspond to the degree of activity in a particular element of a design. The weights ( $W_x$ ) applied to functions give relative values to each function described for a Virtual Prototype. The weights themselves may be functions.

$$\begin{aligned}
 & \text{Equation 1 :} \\
 & f_{Power} = \frac{W_{Pipe}}{W_{Re gAcc}} \times f_{Pipe} + \frac{W_{Instr}}{W_{MemAcc}} \times f_{Instr} + \frac{W_{Cache}}{W_{PeriphAcc}} \times f_{Cache} + \frac{W_{TLB}}{W_{PeriphAcc}} \times f_{TLB} + \\
 & \text{where .} \\
 & f_{Instr} = .2 \times f_{Instr, jmp} + 2 \times f_{Instr, except} + 0 \times f_{Instr, ctrl} + 12 \times f_{Instr, coproc_{15}} + \\
 & \quad 0 \times f_{Instr, LdSt} + f_{Instr, arith} + f_{Instr, other} \\
 & \text{and .} \\
 & f_{Instr, i} = \sum (instructions\ of\ type_i\ in\ k - cycles)
 \end{aligned}$$

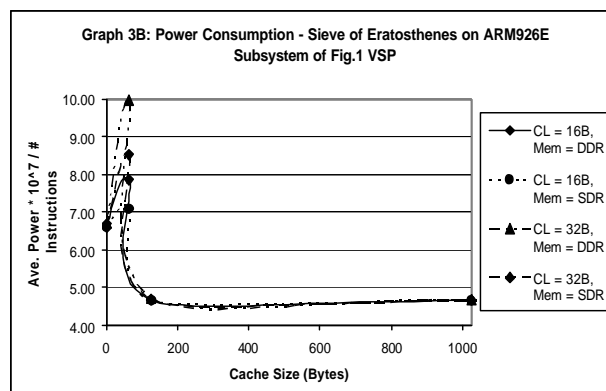
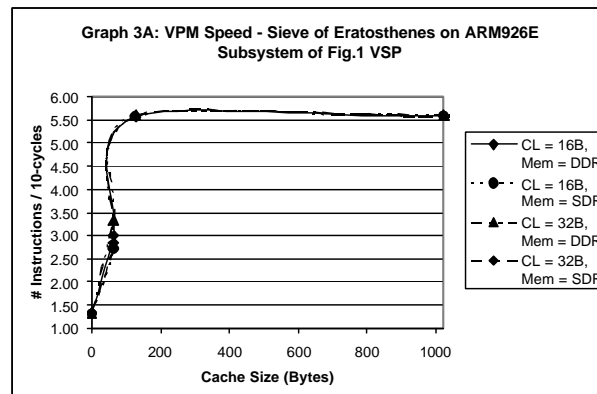
These are the types of functions that drive the quantitative engines of empirical design.

## 5. A Quantitative Empirical Experiment

### 5.1 Optimal Memory Hierarchy

This investigation considered a pure embedded systems problem - that of finding the best tradeoff between speed, power consumption and silicon cost for a small electronic controller executing a limited amount of code – in this case approximated by executing code to solve the sieve of Eratosthenes problem. We were interested in determining the near minimum cache size that would still yield within 5% of optimum speed and power for the controller.

In this experiment we considered a VSP with a single processor having the following instruction and data cache characteristics: size of 0B (uncached), 64B, 128B, 256B, 1 kB, 4 kB and 8kB; cache line size (16B, 32B), wayness (1, 2, 4), cache power weighting (3, 4, 5 – depending on size) and memory types (Double Data-Rate (DDR), Single Data-Rate (SDR)). We varied the relative power consumption of the cache based on size. The results of the experiments are shown in Graphs 3A 'VPM Speed' and 3B 'Power Consumption'. The speed of the VSP followed expectations except that the transition between 64 bytes and 128 bytes was sharp and at 128B the VSP



essentially achieved full speed. The power graph shows another picture. Uncached power consumed by the VSP was 20% - 35% less than the power consumed by 64 byte caches (variability was due to cache line size, wayness and memory type) and 200% higher than power consumed with 128 byte caches. What we are observing here is the step-function effect on power consumption of installing a cache in a processor. For the sieve program, beyond 512 bytes, the power consumed was stable and about 20% higher than the minimum cache configuration at 128 bytes.

The effect on power consumption of installing a small cache in a processor to achieve a 4-fold increase in performance has a detrimental effect on power consumption due to the infrastructure required to support the cache. The cost of a cache is also high since the infrastructure consumes relatively large silicon real estate. These considerations led to an investigation of alternative memory hierarchies that might achieve a better trade-off between speed, power and cost for a controller running a limited amount of code in an embedded application.

We varied the cache\_hit/miss power weightings of the processor to mimic the relative power consumed by a dedicated external buffer of 128 bytes (essentially a small, physically addressed, direct-mapped, on-chip cache external to the processor). This architecture is similar to the buffer organization found in processors like the Renesas SH2A [5] a processor popular in automotive control where differences of cents in the price of a controller translate to several million dollars in large manufacturing runs. The results were that we could achieve a further ~40% power saving whilst maintaining near optimum speed. The cost of the chip is close to the non-cache cost.

This is not an intuitive result and required empirical investigation and experimentation to determine an optimal outcome.

## 5.2 Host Simulation Performance of a Complex System

This is a multi-variable study measuring simulation performance of a host processor simulating the system in Figure 1. This is an example of studies that can be performed to help identify host system architectures that are optimal for supporting the simulations of classes of complex systems.

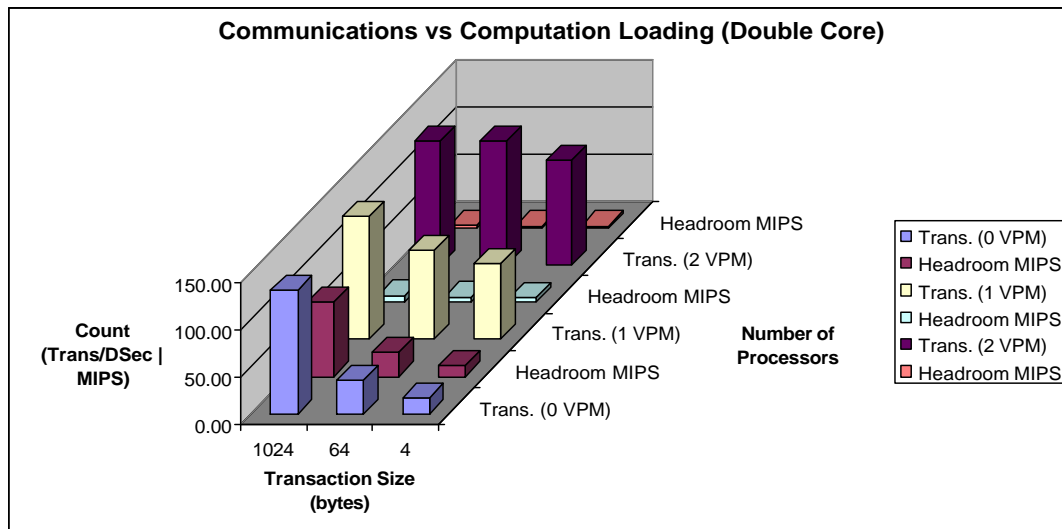


Figure 4. Tradeoff between Computation and Communications Bandwidth

Transactions of various sizes (1024, 64 and 4 bytes) being transmitted at a high rate over a complex switch to which are attached two StarCore SC1200 digital signal processors. Initially no processors are activated and each is then successively activated.

The results bar chart of Figure 4 is best read as a sequence of 3 pairs (Transaction / Headroom (MIPS) – into the slide. As transactions become progressively smaller, there is relatively more work to be performed by the model to transmit and receive

them. The Headroom measure is the amount of available host cycles for further simulation. As more modelled processor are activated and the transaction size is reduced, the available host processing headroom diminishes. If more host CPU cycles were present, perhaps as a dual or quad processing symmetric multiprocessor computing engines, there would provide sufficient cycles to manage high transaction data rates and high levels of target software execution in the simulated multicore networked switch subsystem.

## 6. Summary

Empirical experimentation is a powerful mechanism with which to refute hypotheses that, when carefully constructed, drive the quantitative engineering process. To engage in this engineering process, prior to the existence of a physical realization, requires the existence and use of a model. If hypothesis building concerns speed, power consumption, reaction time, latency, meeting real-time schedules, etc. the model needs to be timing accurate (processor, buses, bus bridges and devices). If the extensive execution of software is an intrinsic part of the empirical experimentation, then the model needs to have high performance across all components. This paper assumes the existence of pre-silicon, high performance (20-100 MIPS), timing accurate virtual system prototypes.

The ability to instrument simulatable system models also helps identify compute engines that are *optimized* for the task of simulating particular types of complex models of systems. An interesting meta-use of the instrumentation of models.

Optimizing systems with complex objective functions is not intuitive. Complex tradeoffs between hardware structure and the software and algorithms that are executed on the hardware cannot be done by conjecture or formal analysis alone, the acquisition of data as part of well-formed experiments refuting thoughtfully constructed hypotheses enables decision making driven by results. Optimization comes from considering the whole system - hardware and software together – not separately.

## 7. References

- [1] Winters, F.J., Mielenze, C. and Hellestrand, G.R. Design Process Changes Enabling Rapid Development. Proc. Convergence 2004 P-387, Oct 2004, 613-624, Society of Automotive Engineers, Warrendale, PA.
- [2] Hellestrand, G.R. The Engineering of Supersystems. IEEE Computer, 38, 1(Jan 2005), 103-105.
- [3] Hellestrand, G.R. Systems Architecture: The Empirical Way – Abstract Architectures to ‘Optimal’ Systems. ACM Conf. Proc. EmSoft2005, Sept 2005, Jersey City, NY.
- [4] Montgomery, D.C. Design and Analysis of Experiments. 5<sup>th</sup> Ed. John Wiley & Sons, NY, 2001.
- [5] Renesas SH-2A, SH2A-FPU Software Manual, Rev 2.00, REJ09B0051 -02000, 13 Sept. 2004, Renesas Technology, Tokyo, Japan.