

Electronic System, Platform, and Package Codesign

Vijay K. Madiseti

Georgia Institute of Technology

Editor's note:

Integrating multicore heterogeneous systems into a system-in-package has challenged many design and test engineers. To overcome these obstacles, we need a common EDA tool for digital, analog, RF, and thermal designs. This article proposes a platform-centric design methodology for modern electronic systems that could incorporate system-on-chip, system-in-package, and system-on-package technologies.

—Yervant Zorian, *Virage Logic*

■ **SYSTEM-ON-CHIP** (SoC), system-on-package (SoP), and system-in-package (SiP) technologies promise efficient, cost-effective, and powerful pathways for realizing complex electronic products. Much of this promise is unrealized owing to a lack of a comprehensive design methodology and supporting tools at the conceptual and specification levels, the early and detailed design levels, and the integration and test levels. The failure to develop such tools stems from the new computation models required for specifying multicore heterogeneous systems and the inability to codesign and test multidomain (analog, digital, RF, and optoelectronic) subsystems concurrently and iteratively. Further contributing to the problem is the lack of a synthesis capability for products' nondigital functions. Although well understood, this problem won't be solved soon, because it involves nearly a dozen areas of design expertise for which designers have few automation tools and even less understanding of their interdependencies.

Current electronic system design-and-test methodologies for communications, wireless, and computing applications can benefit from a migration to a platform-centric approach, which decouples early design from detailed design of product platforms. This methodology maps systems to prefabricated platforms (representing generic templates for implementing domain-specific products), which designers then customize to derive new products efficiently, correctly, and quickly. This platform-centric methodology partitions the design-and-test process into three tasks: representing the specification, selecting and customizing a platform into a product, and

creating library platform objects. Such an approach isolates the experimental nature of platform library creation from in-cycle design tasks.

The problem

New capabilities defining modern packaging have influenced all aspects of the electronic-product design process, changing the way the design is specified, partitioned, implemented, and tested. Designing an electronic package is no longer relegated to the last stages of the design process, prior to integration and test. For instance, as recent product releases by major desktop vendors demonstrate, the change to multicore design affects the application's functional representation (and its underlying computation model) as integrated in the product.

Nonetheless, with all the advances in packaging and SoC technologies, design and test has not advanced commensurately. Cospecification, codesign, cosimulation, and cosynthesis of digital, analog, RF, optoelectronic, software, and passive functionality are proving to be very difficult problems. The conceptual challenge to implementing efficient design methodologies is that digital designs can be synthesized from high-level specifications (for example, from behavior to geometry), whereas most RF and analog designs must be synthesized in the reverse order (from geometry to behavior). Furthermore, codesign of digital and nondigital functions requires redesign of each individual function to ameliorate mutual interference. Figure 1 illustrates the problem facing the design-and-test community: finding the best way to map a system specification to a product using technologies provided by the target system package.

Platform-centric design

Sabbagh defines a platform as a fully functional product family, characterized by a set of commonalities, and

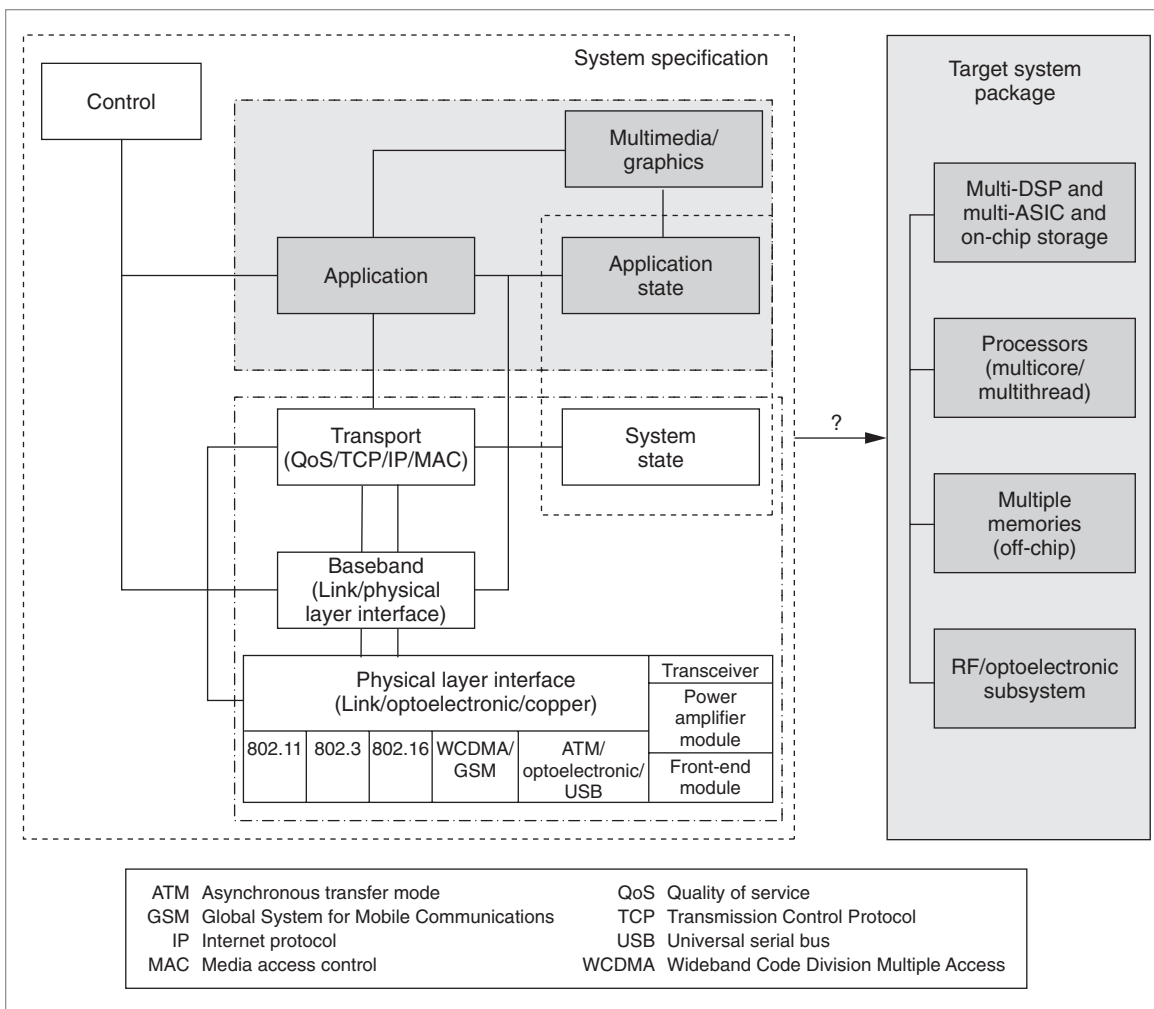


Figure 1. Technical challenge: converting a logical system specification to a product (target system package) using available SoP, SiP, and SoC technologies.

specified and implemented in a way that allows it and its capabilities to be further customized and retargeted for specific end products.¹ Platforms represent a complete system realization, in contrast with IP blocks, which represent portions of a platform; or a PCB-based design template, which represents one instance of a platform.

Examples of platforms are abundant, spanning diverse application areas from aircraft, to automobiles, to PCs, to mobile phones. For instance, Boeing 777 passenger doors, each comprising a different set of parts with subtly different shapes and sizes for its position on the fuselage, are based on the same platform, with 98% of all door mechanisms in common. Another example is PC platforms, which include the x86 instruction set architecture, a full set of buses, legacy support for the interrupt controller, and the specification of a set of communication devices.² In the defense arena, electronic

platforms were called “model year architectures,” as proposed in several DARPA programs in the early 1990s.³

Although commonality in platforms often compromises product performance and hinders innovation and creativity, it expedites the overall product development process. A typical platform can spawn scores of products that are quickly and economically upgradeable through an upgrade of the platform itself. These advantages are highly desirable in today’s system development, in which quick time to market and easy upgrading are the dominating factors.⁷

Figure 2 shows a platform-centric system design approach that can provide a solution to the design-and-test hurdles facing electronic product developers. The time scales for product refreshes are on the order of months, whereas platform refreshes occur more slowly—often over several quarters.

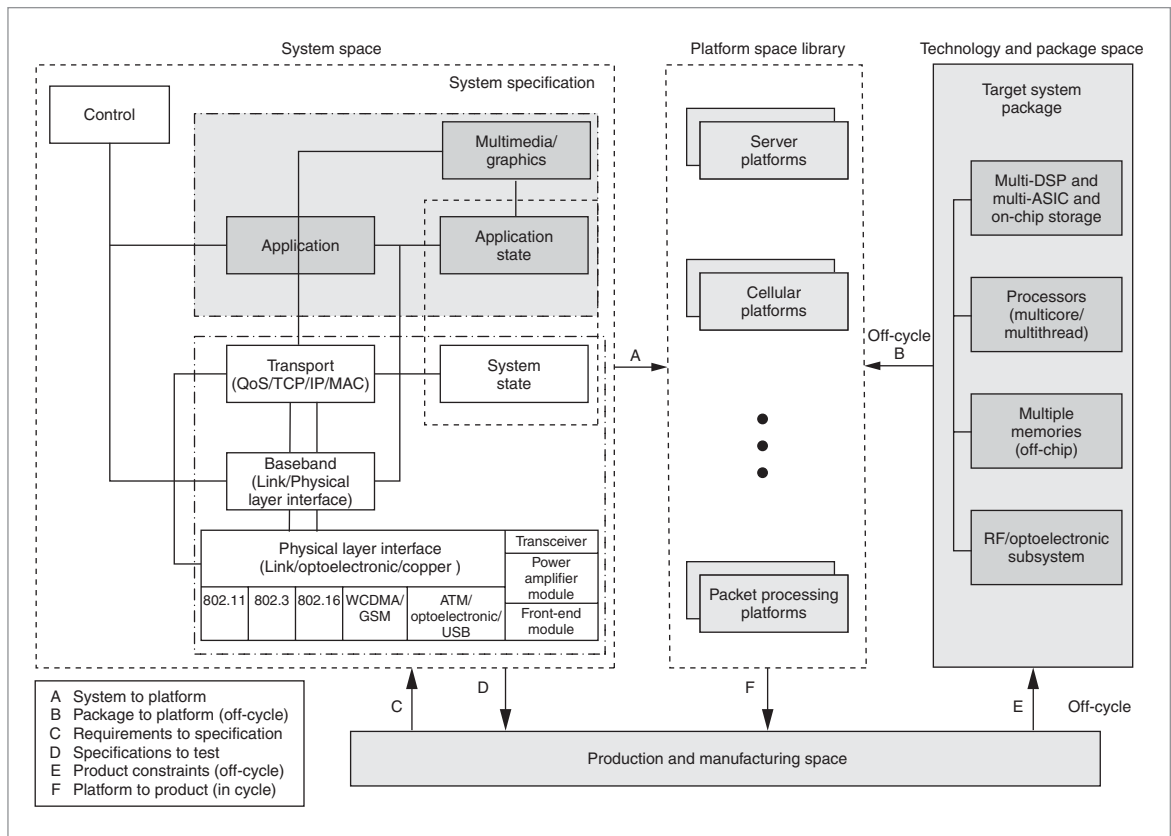


Figure 2. Platform-centric design approach.

It is increasingly evident that chip designers and suppliers are designing application-specific platforms, in contrast with systems integrators, who focus on system functionality requirements, user interfaces, form factors, and marketing. This trend is apparent in wireless and cellular products (TI, Qualcomm, Ericsson), set-top boxes and networking products (Motorola, MIPS, Tensilica, Broadcom, Intel), and PC-based products (Intel, AMD). For these products, the chip supplier provides the eventual system integrators (such as Echostar, Kodak, GE, Cisco, Nortel, Benq, and Siemens) with a complete, customizable, production-ready platform.

There are two general types of system platforms: full and quasi-system. Full-system platforms have complete hardware and software architectures on which designers can implement full applications. They usually consist of a processor, a communication infrastructure, and application-specific blocks. Some also use FPGAs for more flexibility. Examples of full-system platforms include Philips' Nexperia; TI's OMAP multimedia platform; Infineon's M-Gold 3G wireless platform; Parthus' Bluetooth platforms; ARM's PrimeXsys wireless platform; Motorola's Black, Green, and Silver Oak; Altera's Excalibur; Quicklogic's

QuickMIPS; and Xilinx' Virtex-II Pro. Figure 3 shows a simplified view of the TI OMAP platform.

Quasi-system platforms generally don't specify full hardware and software architectures, providing flexibility for system developers to retarget them to a wider range of applications. Platforms such as ARC, Tensilica, and Triscend emphasize the ability to configure processors. Others, such as Sonics' SiliconBackplane and PalmChip's CoreFrame provide neither a processor nor a full application, but rather define interconnect architectures that full systems can be built on.

A system platform of either type also includes tools that help the designer optimize cost, efficiency, energy consumption, and flexibility in mapping an application to the platform.

As Figure 4 shows, the platform-centric approach starts with capturing the requirements and converting them into a functional specification (usually in C, with test vectors and design constraints such as size, real time, and area). The designer then selects one or more platforms that can host the application. The designer captures the platform generically, possibly in a parametric manner, without specifically describing its imple-

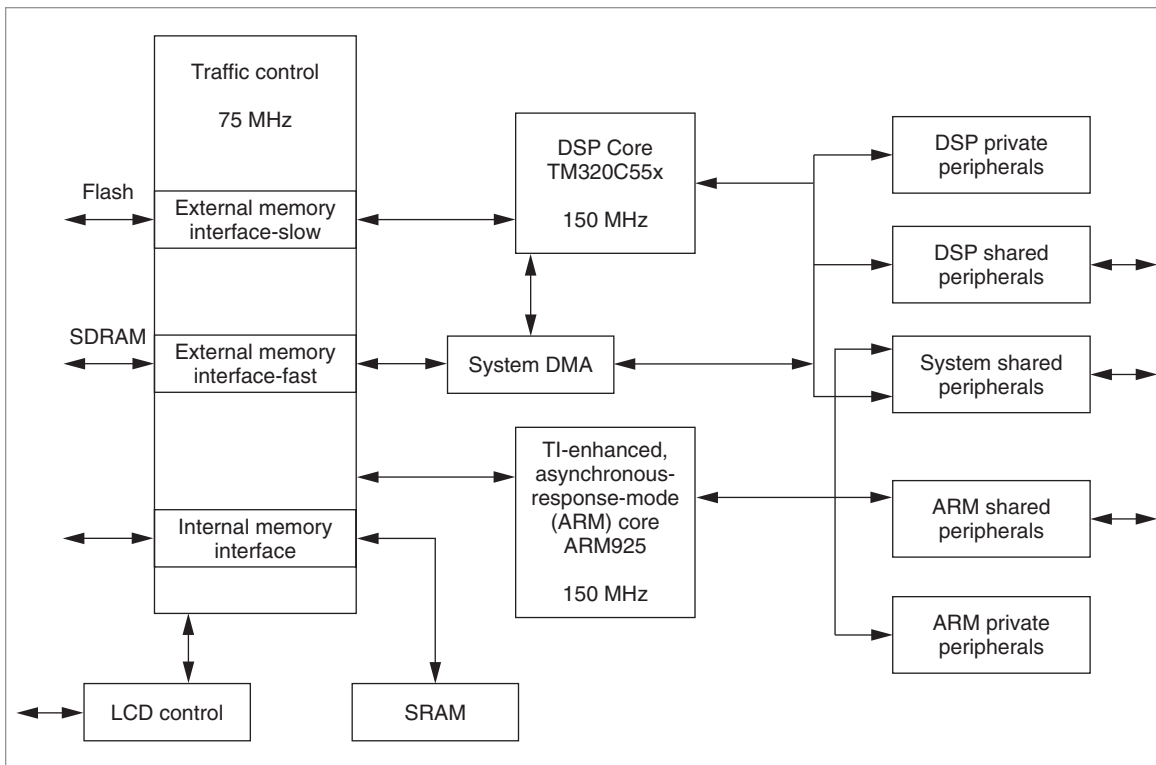


Figure 3. Simplified view of TI's OMAP5910, 289-pin, wireless platform architecture, which has a package size of $12 \times 12 \text{ mm}^2$.

mentation. This allows the use of exploration tools for finding the right fit with the application specification.

After completing platform analysis and selecting a target platform, the designer converts the platform-independent specification to a platform-specific implementation. Figure 5 shows an example of this conversion, which instantiates a wireless-handset platform as vendor-specific components, drivers, and interfaces specified in the original platform in Figure 3.

The designer develops detailed firmware along with the memory layout design and implements the design on the platform prototype—an FPGA-based platform or a demonstration board, for instance.

The original platform developers solve many of the system integration and test design problems offline. Thus, they free the product team from the extensive delays and expense of carrying out these tasks, which typically take 40% to 60% of the total design-and-test effort.

This platform-based approach appears to work well in multiple semiconductor market segments. The approach lets developers quickly launch products with various capabilities and interfaces, while retaining the economy of high-volume production. Figure 6 shows examples of such products.

Recent packaging developments: SoP

Electronics packaging has played two roles in IC devices.⁵ IC or wafer-level packaging has provided I/O connections to and from devices, and systems packaging has interconnected active and passive components on system-level boards. Both tasks were accomplished through interconnections or multilayer wiring at package or board level.

More recently, IC devices began to integrate not only more transistors but also active and passive components on an individual chip, leading the community to believe that someday there would be a single-chip system—a SoC. This development can be called horizontal, or 2D, integration of IC blocks into systems. The IC community began to realize, however, that this approach presents fundamental engineering, investment, computing, and integration limits for wireless and wired communication systems over the long run.

These limitations led to the 3D packaging approach called a system in a package or SiP. But the SiP, though providing opportunities in miniaturization and integration for advanced portable electronic products, appears limited by the CMOS process, just as the SoC does. Some existing and emerging applications, however, include

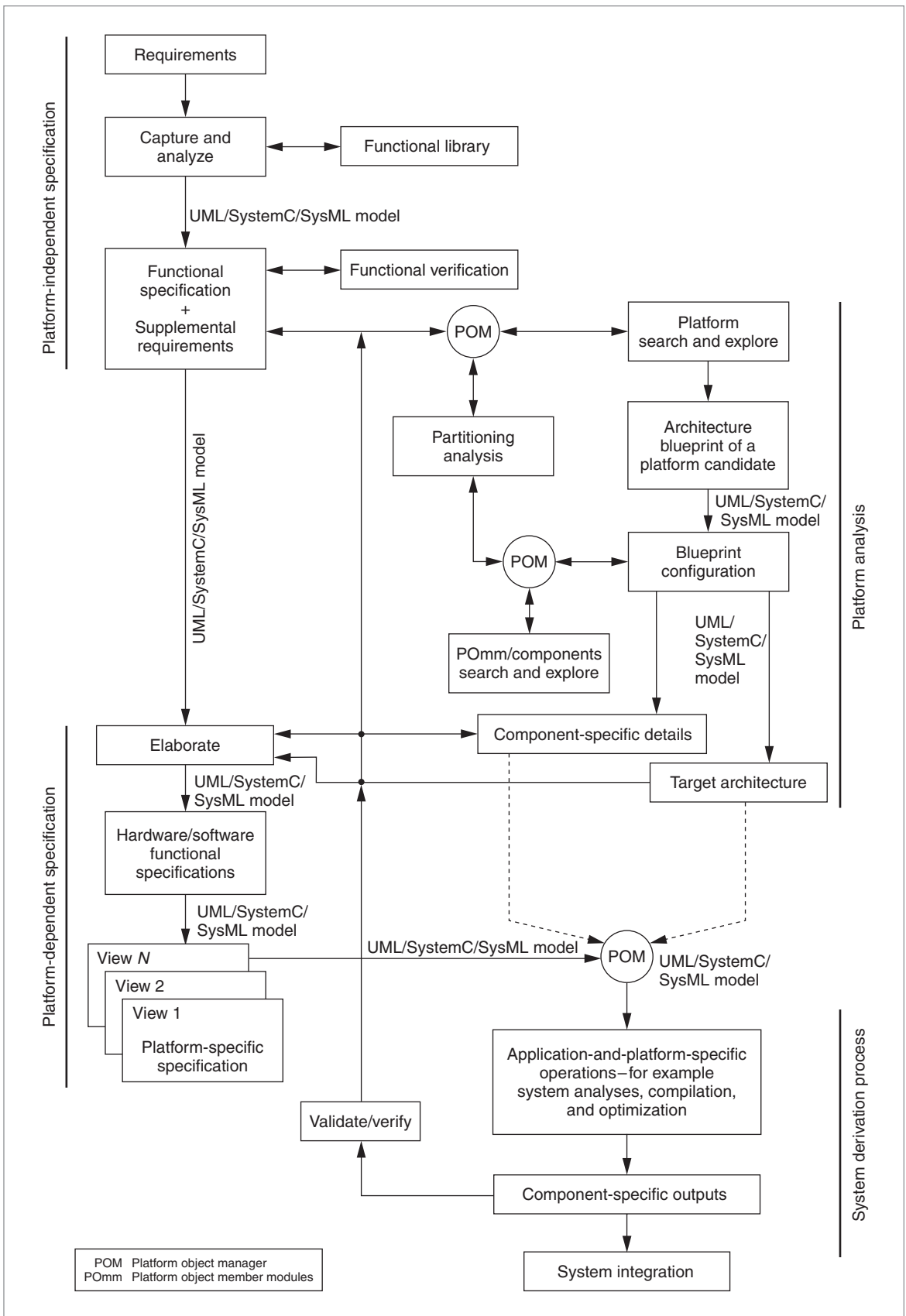
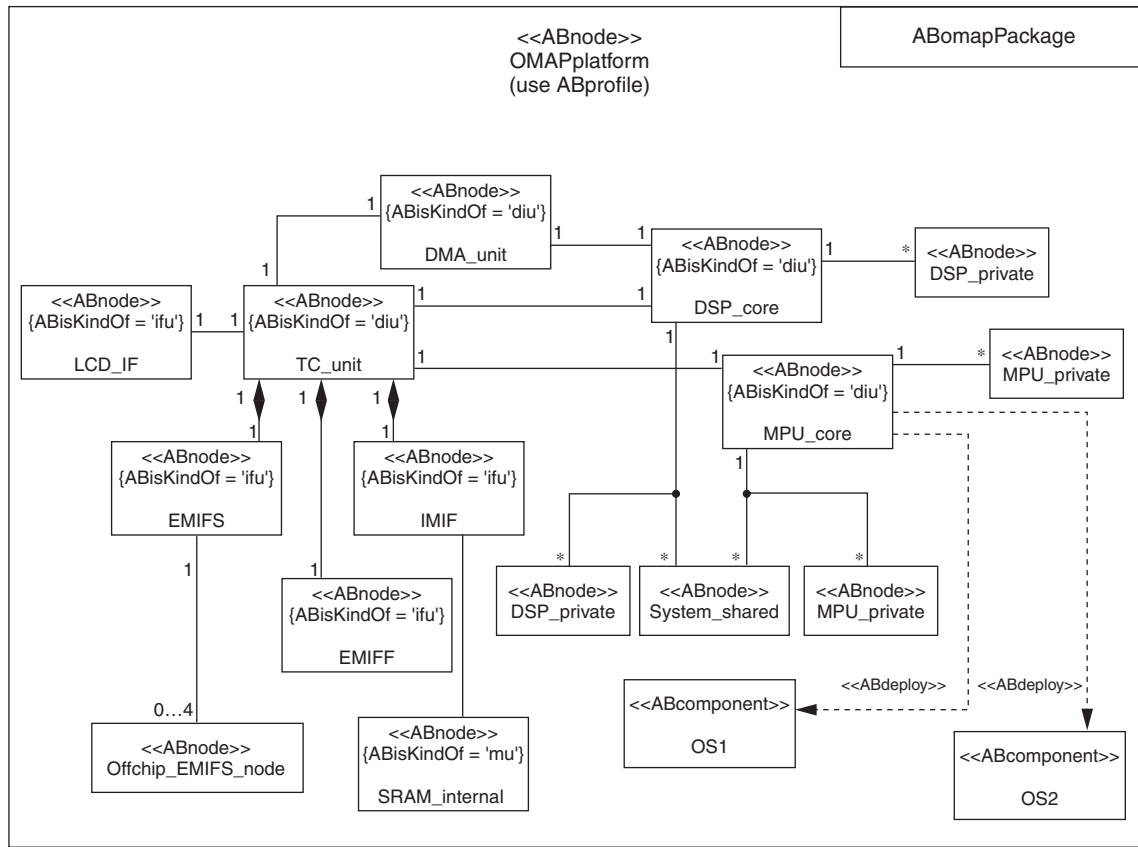
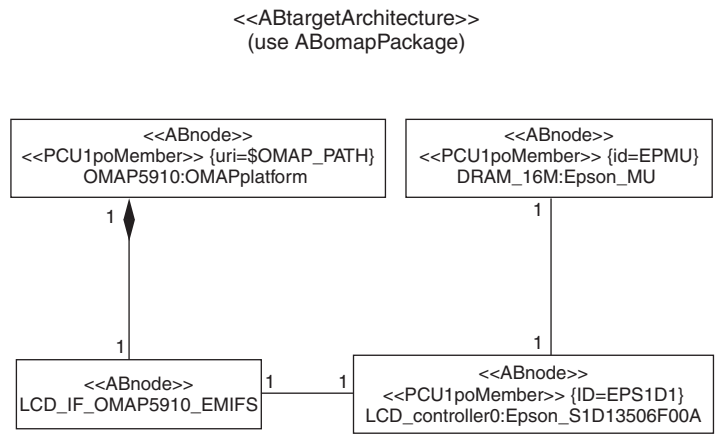


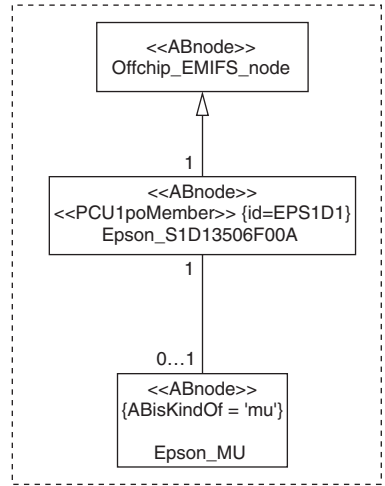
Figure 4. Platform-centric methodology for designing a wide variety of semiconductor products.⁴



(a)



(b)



(c)

Figure 5. Wireless-handset platform’s architecture blueprint, which depicts the abstract representation of the platform architecture (a), and is used by Pomm suppliers as a reference model and by developers to construct the target architecture (b). Each link in the object diagram (b) represents a predefined communication. The DRAM object comes as a derivative requirement when the designer instantiates the liquid crystal display (LCD) controller Pomm module (c).

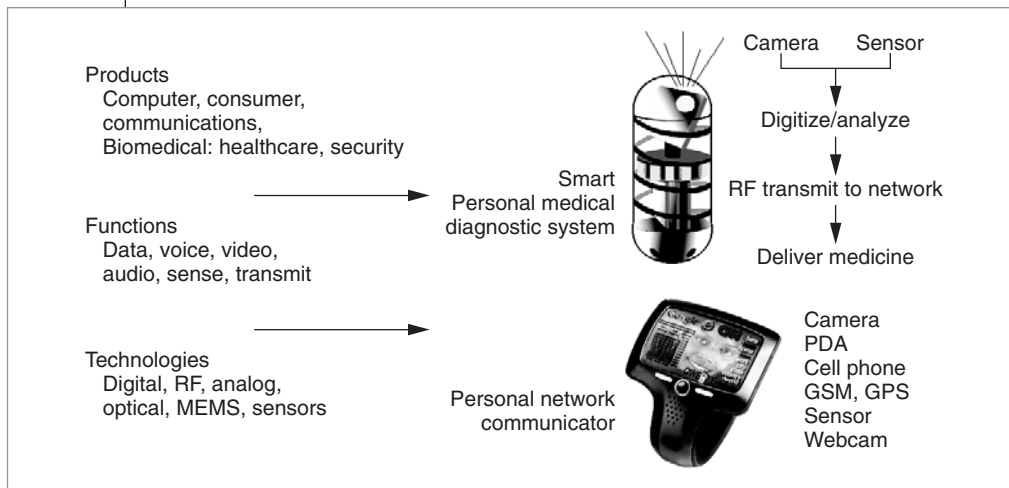


Figure 6. Platform-based miniaturized electronic and bioelectronic systems.

sensors, memory modules, and embedded processors with DRAMs. More recent 3D solutions that incorporate stacked package approaches offer solutions to time-to-market and business impediments that have plagued multichip module (MCM) deployment for the past decade. Cellular handsets have particularly benefited from the SiP approach.

In this section, we quote liberally from the papers written by Tummala who proposed a new paradigm, the system on a package, or SoP, at Georgia Tech in the mid-1990s.⁵ In a SoP, the package, not the board, is the system platform. The SoP addresses the shortcomings of both SoCs and SiPs, as well as traditional packaging, in two ways: It uses CMOS-based silicon for transistor integration, and it uses the package for RF, optical, and digital integration through IC package and system codesign.

SoP packaging, therefore, overcomes the computing and integration limitations of SoCs, SiPs, multichip modules (MCMs), and traditional system packaging. It does this by having global wiring as well as RF, digital, and optical component integration in the package rather than in the chip. A SoP thus includes both active and passive components and functions in a micro-miniaturized platform.

The SoP concept overcomes a number of the SoC's engineering limits. As IC integration moves to nanoscale and wiring resistance increases, the SoC's global wiring delay becomes too high for computing applications. This leads to latency, which designers can avoid either by moving global wiring from nanoscale on ICs to microscale on SoPs or by making digital chips much smaller. The SoP allows both options. The SoP also overcomes the SoC's wireless integration limits as RF com-

ponents such as capacitors, filters, antennas, switches, and high-frequency and high-Q inductors are best fabricated in the package rather than on silicon.

To suppress the expected power plane noise associated with very high-performance ICs, which require more than 100 watts per chip, a major portion of chip area would have to be dedicated to decoupling capacitance alone.

Semiconductor companies are not in the capacitor business; they are in the transistor business. The highest Q factors reported on silicon are about 10 to 25, in contrast to 100 to 400 achieved in a SoP package. Optoelectronics, which today finds use primarily in backplanes and high-speed board interconnects, is moving onto the package for facilitating chip-to-chip I/O and high-speed interconnections. Replacing copper, optoelectronics thus addresses both the resistance and cross-talk problems of electronic ICs. Optoelectronics is not expected to move onto the SoC to replace copper wiring any time soon.

The SoP concept seeks to integrate multiple system functions into a compact, lightweight, thin-profile, low-cost, high-performance system. The system design can call for high-performance digital logic, memory, and graphics, and analog signals for RF and video, as well as broadband optical functions. Unlike a SoC, however, a SoP requires no performance compromises to integrate these disparate technologies, because each technology is separately integrated into the package.

System design times for SoPs are expected to be far shorter, and testing is expected to be simpler. In addition, the SoP concept allows shorter time to market and greater flexibility to take advantage of emerging technologies. The SoP chip size can be small enough for high-yield manufacturing, and its wiring length can be small enough to overcome global signal delays imposed by high resistance.

SiPs can also be regarded as SoPs. In the SiP concept, individual ICs are first packaged and then stacked to form 3D circuits. The individual packages can use the best IC technology and also the best package integra-

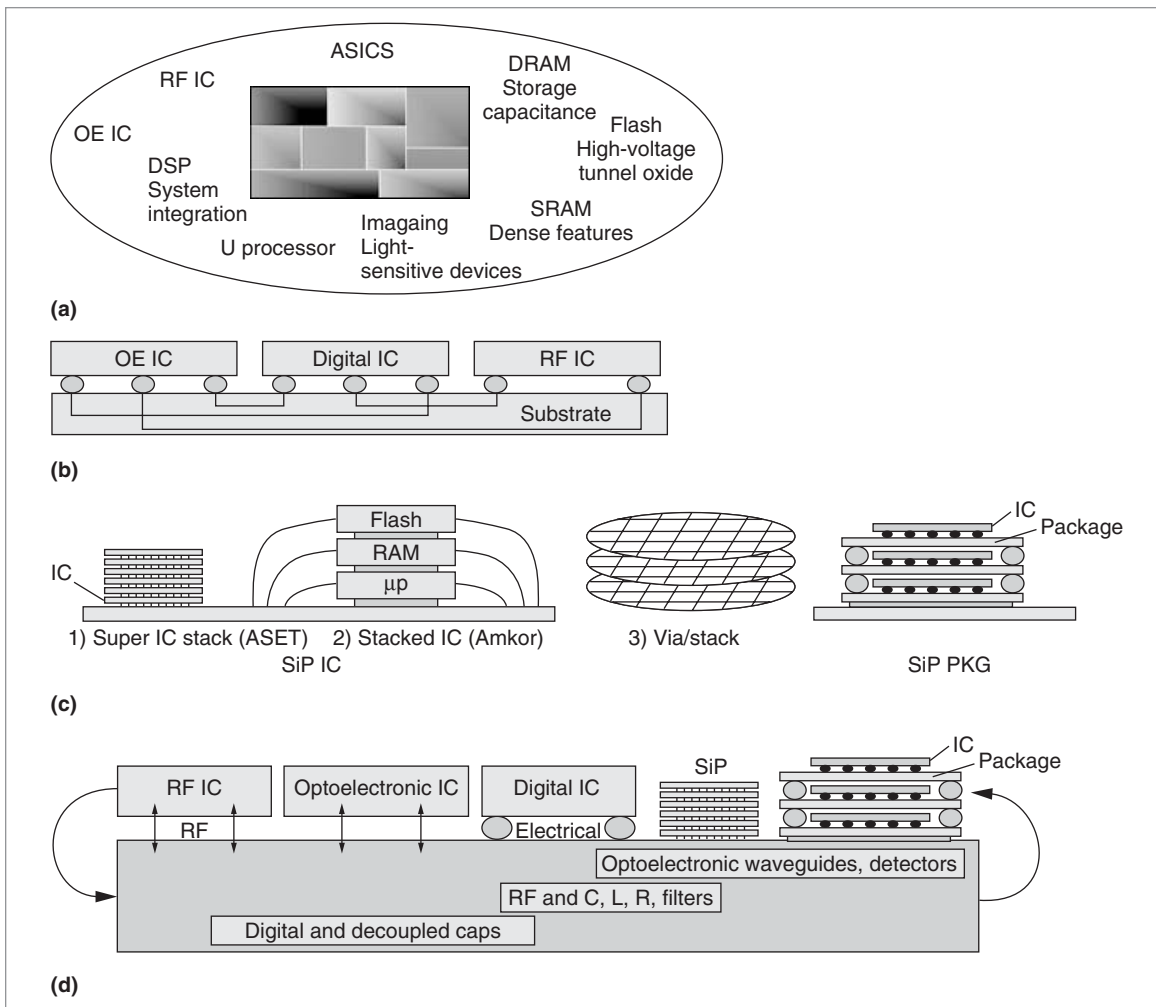


Figure 7. SoC, a complete system on one chip (a); multichip module (MCM) using interconnected components (b); system in package (SiP) using stacked chip or package for reduced form factors (c); and system on package (SoP), which offers the best of IC and packaging technologies by optimizing functions between ICs and the package while miniaturizing the systems (d).

tion technology, such as RF, optical, and certain digital-component integration. However, if the package provides only wiring to interconnect ICs vertically, the package contribution is limited, and the structure isn't a SiP; it is simply a vertically packaged MCM. In fact, such a stack could be more bulky than a SiP. However, such a structure provides partial system integration in addition to power distribution and heat removal.

Figure 7 depicts the integrated-systems packaging evolution, showing the progress from single-chip to MCM, to SiP packaging. The figure divides SiP into two parts, depending on whether the stacking is at the IC level or the packaged IC level, driven by the IC yield. The figure distinguishes SoP from SiP; the former has embedded thin-film component integration in the pack-

age or the board, resulting in potential savings of about 15% to 20% in space and cost.

The SoP and SiP paradigms differ from PCB-based design in several ways. The latter has traditionally been a low-margin electronic manufacturing service, whereas SiP/SoP design looks like a higher-margin semiconductor assembly service. PCB design occurs at the final stages of product design and doesn't usually affect the manufactured product's functionality. In contrast, SiP and SoP require codesign of the application, platform, and package. In addition to the electrical coupling and mechanical structure of PCB-based design, SiP and SoP integrate functionality, such as embedded passives, optical and microelectromechanical systems (MEMS) component integration, and heterogeneous integration of multiple

technologies, that results in 3D design-and-test tasks. These tasks require powerful functional, performance, and multidomain modeling tools. The quality requirements of PCB-based services and SiP- and SoP-based platform services are also very different, with SiP- and SoP-based platform services requiring clean-room technologies.

The SoP paradigm has suffered from a lack of efficient design tool support, from system to implementation. The bottom-up (geometry to behavior) approach followed by recent research experiments in interconnect design has accomplished passive embedding, RF library design, power delivery, packaging-materials design, and thermal-dissipation management. Although impressive, these experiments have resulted in point tools rather than a system design methodology. The platform-centric system implementation approach attempts to overcome this barrier to building efficient systems based on the SoP paradigm.

Challenges in realizing SoP-based products

Design cycles for semiconductor products range from nine to 27 months. Given the complexity of optimized package design, a platform-based design methodology is an economical approach to realizing the large volumes required to reduce packaging and production cost, decrease time to market, and increase productivity.

To bring products to fruition, companies increasingly rely on the systems engineer, a rare breed who is comfortable with system specification, target product requirements, and SoC and SiP integration technologies. Three classes of engineers play a role in implementing a platform-based design methodology:

- *Platform architects* create the generic product template or representation in terms of hardware, software, and interface components, using the best packaging, materials, IC process, and production-related technologies. They design and assemble the SoP platform, integrating heterogeneous digital, analog, optical, RF, I/O, and memory components to create candidates for the library of platform objects. These platforms usually target a particular application domain—for example, cellular platforms, server platforms, network processor platforms, or embedded platforms.
- *System-level design architects* understand the detailed product functionality and interfaces required. They capture the product's functional, form, and cost specifications and select the appropriate SoP platform from a customized template

library. After instantiating the platform and conducting detailed firmware design and customization, they port the application to the platform. They also debug the platform and test it before and after production.

- *Platform module suppliers* provide the building blocks (digital, RF, optoelectronic, analog, software) and process technologies in a reusable form, which the platform architects use to build the generic platform library. They provide the hard and soft IP that constitutes the platform design components. This group includes IP and ASIC component suppliers, memory suppliers, microprocessor and DSP suppliers, communication interface suppliers, and RF module vendors. It also includes technologists, academics, and scientists who develop new technologies for advanced and efficient packaging (ranging from materials and composites research to thermal cooling and interconnect design).

Platform architects and system-level design architects face severe challenges to realizing a process that ensures rapid, cost-effective system-level design. Platform module suppliers' task is less difficult, because they rely on standardized interfaces to the system-level design platform's logical objects.

Issues facing platform architects

Figure 8 shows the platform architect's typical design process and associated tasks. The platform architect usually constructs the platform from the following design objects: digital functions, analog functions, RF functions, and, in higher-performance products, optoelectronic functions. Designing an efficient platform requires detailed knowledge and tools in nearly a dozen areas of multidisciplinary expertise, and designers can seldom complete this task during the design cycle in current practice. The package, which provides interconnection and integration, provides additional functionality through passive components (resistors and inductors) that can be embedded into its substrate. In the area of digital functions, the platform architect makes additional choices as to the control processor, the DSP functions, and the storage (memory) hierarchy.

Digital functionality

The digital functions that the platform architect works with include multicore digital architectures, memory architectures, and power distribution.

Multicore digital architectures. The *International*

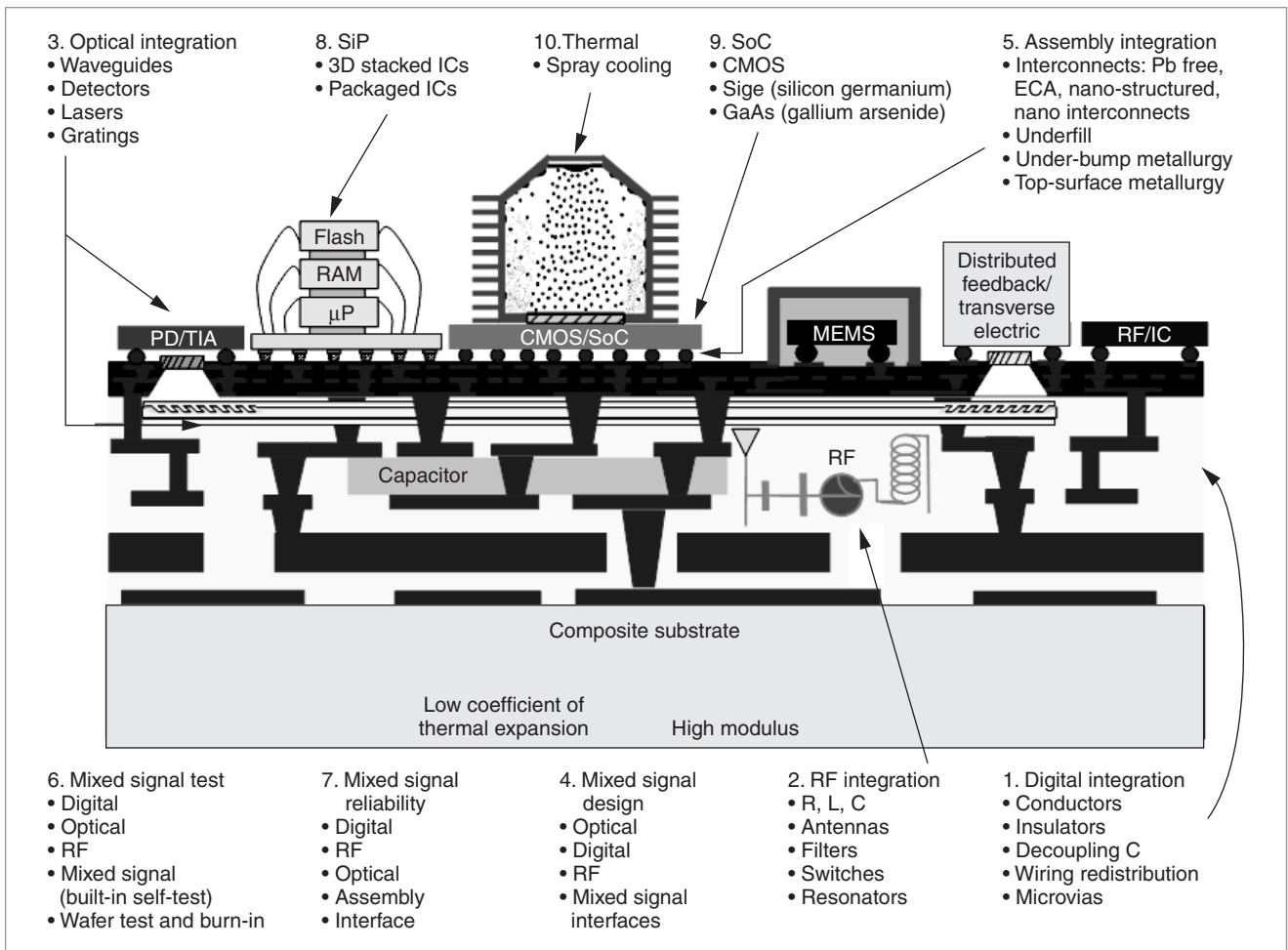


Figure 8. Platform architect's job: building an efficient platform for semiconductor products based on heterogeneous technologies and inter-disciplinary design skills.

Technology Roadmap for Semiconductors (ITRS) shows that increasing clock speed is not the best way to gain performance, whether in a cellular platform, a cost-performance chip, or a high-performance server processor. Indeed, the roadmap predicts that clock rates will hit a power barrier between 5 and 10 GHz, and a better use of transistors would be multicore processors running at far slower clock speeds. Even if scaling allowed an increase in clock frequency, system-level considerations limit the pipeline stages to four to six for DSP and multimedia applications, and eight to 14 stages for higher-performance applications. Performance is expected to saturate at about four FO4 gate delays. There has been much research in electronic packaging that targets interconnection speeds of 10 GHz or higher. Application of such packaging is likely to be very limited, because lower-speed multicore approaches can sustain the performance increase needed for high-performance, cost-

performance, and embedded-processor applications in the digital domain.

An additional packaging challenge is to reduce the power dissipated during I/O transfer to less than 5 mW/(GHz)(bit) for sustained data transfer rates for high-performance applications. As discussed by Flynn and Hung,⁶ the relationships of area (A), power (P), and time-performance (T)—that is, $PT^3 = \text{constant}$, and $AT^2 = \text{constant}$ —will continue to drive digital processor design, as opposed to the ITRS prediction. Additional arguments related to high-speed interconnect design that argue against increasing clock frequencies include the increased negative effects of incompletely modeled electromagnetic interference, reflections, and cross talk (signal integrity).

Memory architectures. High-performance and cost-performance applications will continue to use external

memory but will require efficient cache hierarchy design to hide the delay caused by the interconnect. As a result, caches will dominate the digital-subsystem area, occupying more than 80% of on-chip silicon.

For mobile and low-power platforms, use of on-chip memory will continue, although stacked SiP solutions will be just as effective for both performance and cost. Also, SoCs will increasingly use SRAM, ferroelectric RAM (FRAM), magnetic RAM (MRAM), and ovonic unified memory (OUM) for on-chip solutions. These memories require less area and a few (typically, two or four) additional masks for CMOS processing in 90-nm or lower technology.

Power distribution. So far, active power has been the main consideration in CMOS power delivery. As CMOS scales below 90 nm, and V_{DD} values drop, device leakage currents will contribute to passive power. Passive power will continue to increase and put additional strain on the on-chip power distribution system, as passive power further diminishes the DC drop noise budget. Platform architects must carefully analyze instantaneous voltage-drop variations and current demands and design suitable on-chip decoupling capacitors (with capacitance densities on the order of 500 nF/cm² to 1 μF/cm²). Efficient power management algorithms that activate and deactivate logic islands add to the distribution challenges. This is an area of active research.

RF functionality

RF functions available to the platform architect include RF radios and embedded passives.

RF radios. Cellular radio platforms consist of three main functional blocks: transceiver, power amplifier module, and front-end module. The power amplifier and front-end modules are implemented as separate packages, to prevent the power amplifier module's components from thermally affecting the front-end module's sensitive filters.⁷ The platform-level design choice is whether the transceiver should be integrated with the baseband processing on the SoC. Advantages of including the transceiver on the SoC include eliminating expensive passive components and using the high sampling rates available in digital CMOS technology. Integration on the SoC also improves yield and reduces test costs.

Embedded passives. Embedding passive components (up to several hundred) in the substrate can decrease the area and cost of RF modules that use R , C ,

and L components by 25% or more, although it might increase the processing time per board in the production process. Architects commonly use embedded capacitors in cellular platforms because they have a high passive density and can be easily embedded, and also because higher- Q (> 60) inductors are difficult to integrate in a CMOS IC.

Platform architects can choose from several technologies to implement embedded passives: laminates, multilayer ceramics, and organic liquid-crystal polymers. Recent developments in thin-film RF materials make embedded passives particularly relevant to wireless platforms. A 3D design approach leads to high quality and efficient realization of embedded passives that can support multiband and high-bandwidth requirements. Typical design accuracies of higher than 5% for embedded passives make it possible to construct resonators, impedance-matching networks, filters, and transformers required for embedded applications.

The main problem in relating embedded passives to system-level design is not whether they are feasible, but whether they are economical in terms of the product life cycle. Life-cycle costs include design-and-test costs, non-recurring costs (for new equipment), time-to-market costs (additional design, verification, and test, and lack of reuse across multiple wireless communication modes), and ease and cost of reworking. Sandborn presents a detailed analysis of the relative advantages and production costs of systems with embedded passives.⁸ According to other reports, the advantages of embedded passives in size and cost reduction for handheld wireless and portable products appear to be compelling.

Analog functionality

Analog functionality, primarily in cellular platforms, takes the form of analog and power management circuitry. The platform architect can integrate analog functionality into the CMOS SoC, subject to the following constraints: The circuitry must be redesigned for suitable implementation on low-power digital CMOS logic. The redesign offers several advantages, including the use of low-power, multibit sigma-delta converter architectures and low-power comparator implementations. Design isolation is also necessary to shield the analog components from the digital clock noise. Some powerful isolation schemes, such as adding additional buried n++, require additional processing steps (and are more costly).

Issues facing system-level design architects

In capturing the product's specifications and select-

ing the appropriate SoP platform, system-level design architects' main problems are selecting the multicore computation model, migrating application code across platforms, and code generation and debugging.

Multicore target and computation model. Application developers commonly use high-level languages such as C, or application-specific languages such as Matlab, to represent complex product functionality and its expected behavior.

It is apparent that there is a mismatch in the underlying computation model between the way applications are commonly represented in sequential and imperative languages such as C and multicore SoC platforms (concurrent platforms with various synchronization mechanisms). System-level design architects are responsible for the design-and-test of dataflow and control flow onto a concurrent platform, a paradigm shift in the system specification process.

The multicore model is different from the massively parallel approaches in vogue in the early 1990s. It is related more closely to the multithread, coarse-grained parallel models suitable for server and multimedia applications or the multicore, pipelined, accelerator-assisted models of platforms used in wireless and communications applications.

System-level design architects also use architectural-level exploration tools to find the right partitioning between control software, DSP functionality, and application-specific accelerators required for meeting system-level specifications.

The platform model supplied to system-level architects is usually a simulation model captured in a language such as SystemC, with support for underlying time- and event-driven semantics, and various computation and synchronization models. Models for accelerators and IP cores obtained from IP suppliers can also be integrated in the SystemC-like framework. The OpenMax initiative (<http://www.khronos.org>) is an example of the standards-driven approach to defining specifications and models for accelerating system-level applications through disciplined accelerator use.

Figure 9 shows an example of such a framework—a multicore compilation suite (MCS) architecture. The

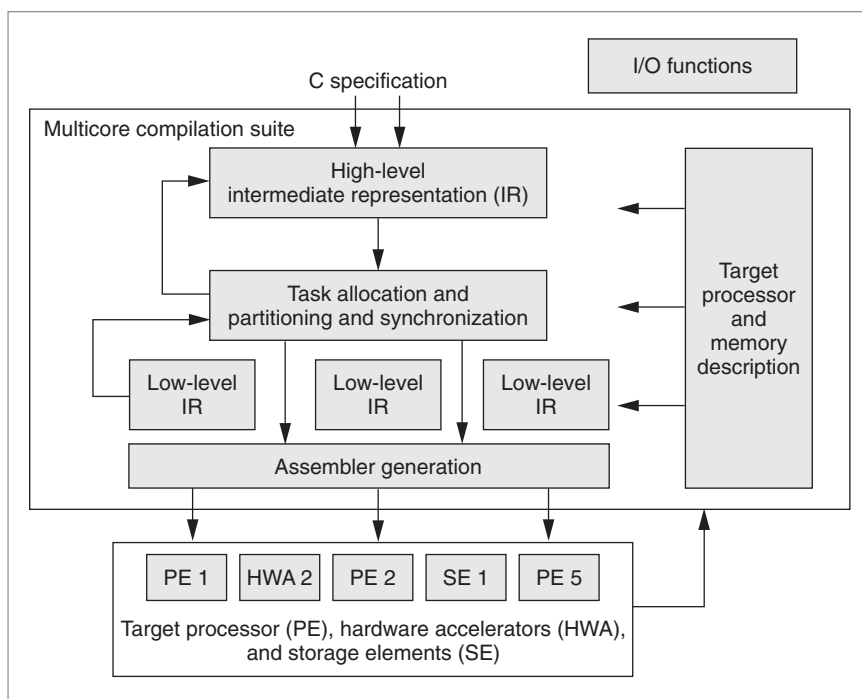


Figure 9. Overview of multicore specification (MCS) code generation process. Packaging technologies are driving system-level architects to adopt new models of computation through multicore specification and compilation suites.

system-level architect converts the original C specification of the design (which includes digital, RF, analog, communications, and storage functions) to a variation of C that supports concurrency, multiple threads, hardware abstractions, and synchronization and control primitives. Proposed languages, such as SystemC and SysML, are similar to this variation. The MCS code-generation framework provides a detailed description of the platform's architecture, interconnect, timing behavior, and interfaces. The architect converts the C specification to a higher-level intermediate representation suitable for mapping to a heterogeneous architecture (consisting of digital, analog, RF, and storage components). This partitioning results in a low-level intermediate representation that the architect uses as follows:

1. Passes code partitions mapped to software through compilation tools for mapping to processors.
2. Uses code mapped to hardware accelerators or ASICs for testing functionality and interfaces.
3. Uses synchronization primitives to generate control code that glues the heterogeneous components together.

Platform-centric design guidelines

These six guidelines can help platform and system-level architects perform their tasks efficiently.

- *Design guideline 1.* In the next five years, clock rates will saturate in the range of 5 to 15 GHz, despite higher predictions of the *International Technology Roadmap for Semiconductors* (ITRS), and platform architects will be able to extract performance from a multicore, multi-thread approach at far lower clock frequencies. These architects will increasingly use special-purpose hardware accelerators (such as a Discrete Cosine Transform module for video decoders, and TCP/IP offload engines for packet processors) for their positive effect on power, performance, and cost.

- *Design guideline 2.* Cache memories will dominate chip area for high-performance applications and can consume 80% or more of die area as compared to logic. High-performance processors will continue to use larger off-chip memories for storage lower in the memory hierarchy. For lower-cost embedded solutions such as cellular platforms, memories will continue to be included on chip, and newer non-volatile memories, such as ferroelectric RAM, magnetic RAM, and Ovonic Unified Memory, could see increasing use.⁷ However, the cost of each memory cell on the chip is about an order of magnitude (10 to 20 times) higher than the cost of a memory cell on a standard memory chip. A flip-chip memory attachment to

4. Maps common data structures to storage areas, and uses a stream manager to schedule data streams flowing from storage to computational units (ASICs and processor registers).
5. Develops detailed test vectors for each subsystem for subsystem-level verification and system integration.

Application code migration. The platform-centric approach often results in a product refresh cycle of weeks or months, whereas the platform refresh cycle can take several quarters or years. Thus, system-level architects need efficient tools for migrating designs ported to one platform to the succeeding one, while keeping legacy functionality intact. Since much of the code and firmware developed for embedded and cost-performance platforms is written by hand (optimized for the underlying hardware and operating system), automated methods for migration should gain much commercial interest in the near future.

Code generation and debug. Once detailed firmware is generated, the next important step is integration and test of the entire system. The amount of debugging that is necessary because of the parallel and concurrent nature of the mapped application complicates the integration process.

In multicore designs, processors communicate with each other through specialized semantics, using either shared registers or buses, and through memory.

Constructs such as semaphores serialize access to common data to preserve computation correctness. Errors can result from access ordering and from incorrect setup of multicore synchronization and communications conditions.

To facilitate multicore debugging, system-level architects might need to insert additional test logic into the processors and cores using a JTAG-like extension. A recently developed standard called Nexus 5001 (or IEEE-ISTO 5001) extends JTAG to insert on-chip debugging logic, using a similar packet-based protocol as an option. Nexus 5001 also includes a faster approach using auxiliary ports configured for full-duplex operations. Whereas JTAG provides static examination of the processor, Nexus provides examination of the processor running at full speed, allowing dynamic analysis of its behavior at the cost of the auxiliary port. The amount and type of debugging information that designers can collect via Nexus 5001 is classified into multiple levels, so that designers can make tradeoffs between transfer rates, information types, and cost. Nexus specifications also allow multiple cores to share an auxiliary port (to save pins).

The “Platform-centric design guidelines” sidebar suggests ways for platform and system-level architects to perform their tasks efficiently.

AN INTEGRATED APPROACH to codesign of system, platform, and package will result in rapid insertion of new

processors or to an ASIC appears to be a good compromise.

- *Design guideline 3.* The platform architect's main design decision is whether to include the RF transceiver as part of the baseband SoC. The architect will usually implement the RF radio's front-end and power amplifier modules as separate packaged modules. Platforms incorporating embedded passives will benefit significantly from the platform-based design approach. In current practice, the bottom-up design approach to constructing a library of embedded-passive components and experimentally integrating them in a reference design can lead to lengthy design cycles.
- *Design guideline 4.* Integration of analog with digital CMOS is well understood technically and attractive from a cost viewpoint. It is rou-

tine in most cellular platforms and is the preferred approach for low-cost, large-volume products.

- *Design guideline 5.* Multicore compilation, simulation, and code specification and generation environments represent a revolutionary shift for the future system architect. Multicore Compiler Suite tools are critical to efficient and cost-effective mapping of systems to platforms, and their instantiation as products.
- *Design guideline 6.* Debugging at runtime within a complex multiprocessor environment is one of the more challenging and expensive tasks of product and platform design. Thus, debugging standards that are based on dynamic analysis standards, such as Nexus 5001, can be important enabling technologies for multicore system-level design.

technology into platforms and subsequently into the next product generation in a timely, predictable, and cost-effective manner. The platform-centric approach requires a new type of engineer, the system-level design architect, or more simply the integrated-systems engineer. ■

■ References

1. K. Sabbagh, *Twenty-First Century Jet: The Making and Marketing of the Boeing 777*, Scribner, 1996.
2. A. Sangiovanni-Vincentelli and G. Martin, "Platform-Based Design and Software Design Methodology for Embedded Systems," *IEEE Design & Test*, vol. 18, no. 6, Nov.-Dec. 2001, pp. 23-33.
3. V.K. Madiseti and A.J. Gadiant, "Model Year Architectures," *The RASSP Digest*, vol. 2, qtr. 4, 1995; http://www.eda.org/rassp/documents/newsletter/html/95q4/news_2.html.
4. V. Madiseti and C. Arpnikanondt, *A Platform-Centric Approach to System-on-Chip Design*, Springer, 2005.
5. R. Tummala, G. White, and V. Sundaram, "SoP: Microelectronics System Packaging Technology for 21st Century: Prospects and Progress," *Proc. 12th European Microelectronics and Packaging Conf.*, 1999, pp. 327-335.
6. M.J. Flynn and P. Hung, "Microprocessor Design Issues: Thoughts on the Road Ahead," *IEEE Micro*, vol. 25, no. 3, May/June 2005, pp. 16-31.
7. W. Krenik, D. Buss, and P. Rickert, "Choose the Best IC Integration Method When Designing a 3G Handset," 2005, [http://www.us.design-](http://www.us.design-reuse.com/articles/article10472.html)

[reuse.com/articles/article10472.html](http://www.us.design-reuse.com/articles/article10472.html).

8. P.A. Sandborn, "Economics of Embedded Passives," *Integrated Passive Component Technology*, R. Ulrich and L. Schaper, eds., Wiley-IEEE Press, 2003.

Vijay K. Madiseti is a professor of electrical and computer engineering at Georgia Institute of Technology. His research interests include DSP, computer engineering, system-level

design methodologies and tools, and software systems. Madiseti has a BTech in electrical and computer engineering from the Indian Institute of Technology, Kharagpur, India, and a PhD in electrical engineering and computer science from the University of California at Berkeley.

■ Direct questions and comments about this article to Vijay K. Madiseti, Georgia Institute of Technology, School of ECE, Mailstop 0250, Atlanta, GA 30332; vkm@ece.gatech.edu.

For further information on this or any other computing topic, visit our Digital Library at <http://www.computer.org/publications/dlib>.