

EDA360

The Way Forward for Electronic Design

"As founder of an organization that is transforming the economics and technology of an industry so that we can improve opportunity for millions, I am constantly watching what others are doing. Cadence has a vision and model with the potential to transform the economics and performance of the microprocessor industry and, by extension, devices every consumer takes for granted. This document outlines a vision that bears watching."

– Nicholas Negroponte, Founder and Chairman, One Laptop per Child

"The changing basis of competition in some mainstream semiconductor markets will punish firms that do not adapt to the new environment. EDA360 lays out a high-level framework, leveraging modularity and architectural innovation, for profitable adaptation to this new wave of disruptive change."

– Steven King, Founder, Innovo Strategies

"Today's 1.5B connected consumers will balloon to 6B or more by 2015—but those consumers will join the global network in emerging markets, where factors like income, literacy, and access to electrical power require a new set of breakthroughs in the electronics industry. The EDA360 document issues a call to action for the semiconductor industry to deliver these breakthroughs."

– Yankee Group's Emily Green, author of "Anywhere: How Global Connectivity is Revolutionizing the Way We Do Business"

"Being able to design complete systems from the ground up and deliver a total platform for system integration, applications development, and system validation represents the future of EDA. The EDA360 vision has the potential to allow semiconductor companies to create powerful products that would enable consumer technology providers to deliver a device platform with an ecosystem of hardware, software, and services. That in itself could help tech companies be more competitive and profitable."

– Tim Bjarin, President, Creative Strategies, Inc.

EDA360: THE WAY FORWARD FOR ELECTRONIC DESIGN

Chapter 1: EDA Industry Focus Shifts to Integration and Profitability	1
Chapter 2: Application-Driven System Realization	8
Chapter 3: Software-Aware SoC Realization	16
Chapter 4: EDA360 Enables Silicon Realization	22

CHAPTER 1: EDA INDUSTRY FOCUS SHIFTS TO INTEGRATION AND PROFITABILITY

INTRODUCTION – A REVOLUTION INTERRUPTED?

From mobile phones to infotainment devices in cars, from digital cameras to heart monitors, from e-readers to network-aware televisions, our world is shaped by increasingly sophisticated and interlinked electronic devices. A world without devices that were inconceivable just 10 or 20 years ago would be intolerable today. The systems and semiconductor companies that shaped this electronics revolution have not only increased functionality exponentially, but have also slashed costs to the point where a few dollars today purchases more computing power than millions of dollars 30 years ago.

While we all recognize names such as IBM, Intel, Samsung, Apple, and Sony as leaders of this electronics revolution, it's important to remember that this era of innovation has been empowered by electronic design automation (EDA). The commercial EDA industry provides the software, services, and intellectual property (IP) that make advanced semiconductor and system design possible. Several decades ago, EDA software was internally developed and was used only by a few large companies. Over the past 30 years, the commercial EDA industry has brought advanced integrated circuit (IC) design capabilities to companies of all sizes, including startups that have developed some of the most creative products in the market.

Today, systems and semiconductor companies are undergoing a disruptive transformation so profound that even the best-known companies will be impacted. The EDA industry now stands at a crossroads where it also must change in order to continue as a successful, independent business. Without that change, EDA will become a fragmented industry offering suboptimal, poorly targeted solutions that fail to solve customer problems. As a result, the huge leap forward provided by the electronics revolution will come to a standstill. The result? A squandered opportunity for technology innovation, and a diminished contribution by the electronics industry to re-build the global economy.

The disruptive transformation we are speaking of is not about EDA developing new design tools. It is not about new methodologies. It is not about the functional verification crisis, or the move to electronic system level (ESL) design, or any of the issues that have dominated discussions about EDA to date. It is about something much larger. It begins with a shift from design creation to integration in the electronic systems industry, and results in a new focus on profitability. This realization, in turn, opens the way to EDA360, a new vision for what the EDA industry can become.

To tell the story, however, we must take a step back for a moment to where everything starts—the electronics consumer.

THE “NEW” CONSUMER ELECTRONICS MARKET

EDA providers have long delivered value to computer-aided design (CAD) departments and engineering teams, but have often failed to successfully articulate a value proposition to corporate leaders. To do so, EDA providers must fully understand the market imperatives that electronics companies are facing, and help those companies deliver value to their customers. End consumer demands help shape EDA technology development. For example, the demand for low-power devices and long battery lives has led to a growing suite of low-power IC design tools.

So what do electronics consumers want? Their appetites are voracious and growing. As of now, the most prevalent demands include:

- **Multiple, concurrent software applications.** Consumers want to make phone calls, watch movies, find directions, text friends, and surf the Web—all on the same mobile device.
- **Complete mobility.** Consumers want to connect to the Internet anywhere, anytime. They want lightweight, portable devices with long battery lives. And they want their experiences with these handheld devices to be seamless, requiring a high level of integration among device, software, and network.
- **Audio, video, 3D graphics.** Text and voice only are no longer sufficient. We live in a visually-oriented world where high-resolution video and graphics are becoming an essential part of any electronics product.

And of course, consumerization coupled with today's economy drives further cost pressure. That is especially true given the large number of potential consumers and producers in China and India.

Today, systems and semiconductor companies are undergoing a disruptive transformation so profound that even the best-known companies will be impacted. The EDA industry now stands at a crossroads where it also must change in order to continue as a successful, independent business.

Established companies in the systems design space have largely responded to these challenges by making the same kinds of incremental improvements they've made all along. They have continued to call upon their semiconductor partners for faster, better, lower-power ICs, but most have not changed their fundamental business models. And today, established electronics companies are being challenged by game-changing new entrants.

Many of the newcomers are redefining their industries around applications, or “apps.” For example, in the mobile handset market, established companies such as Nokia, Samsung, and Motorola are competing with newcomers including Google (Android), Apple (iPhone), HTC, and RIM. In the mobile infrastructure market, companies such as Alcatel/Lucent, Nortel, and Ericsson are facing new entrants including Huawei and ZTE. And in the PC market, Asus and Acer are challenging well-known players such as Dell and Hewlett-Packard.

Applications take different forms in different industries. “Apps” in mobile handsets could include audio, video, gaming, and the myriad of applications available in the iTunes Store.

“Apps” in mobile infrastructure could include routing, deep packet inspection, and encryption/decryption. And this “apps” trend extends to virtually every vertical industry. The significant point is that most of the newcomers are focusing their innovation and differentiation on “apps.”

Many of the newcomers are outsourcing hardware design. In addition, they are increasingly dependent on semiconductor providers to supply portions of the software stack, such as diagnostics, drivers, operating system (OS), and middleware. What they are demanding, in effect, are application-ready platforms with hardware and software for a given application, such as mobile computing. The completeness and relevance of an application-ready platform has become as important as having the latest, greatest, most power-efficient silicon. The newcomers differentiate their products by building unique software applications on top of those platforms.

There are several approaches to building application-ready platforms. In a traditional, disaggregated development approach, hardware is developed first, and the OS and applications are added later. In a hardware-defined approach such as the Apple iPhone, the hardware and OS are fully integrated, but applications must still conform to the hardware. In a hardware-independent, application-driven development model—exemplified by the open-source Google Android OS—hardware can be built to meet the needs of the application. We will discuss these approaches in more detail in Chapter 2.

FROM CREATORS TO INTEGRATORS

Semiconductor makers, meanwhile, have been chasing Moore’s Law for the past three decades. This “law” holds that the number of transistors that can be placed on an IC doubles every two years. It depends on a continuing migration to lower process nodes to gain performance and die size advantages. However, Moore’s Law is hitting a wall due to rising development costs. According to the analyst firm Semico, system-on-chip (SoC) development costs at the upcoming 32nm process node will approach \$100 million. Much of this cost will be SoC software development.

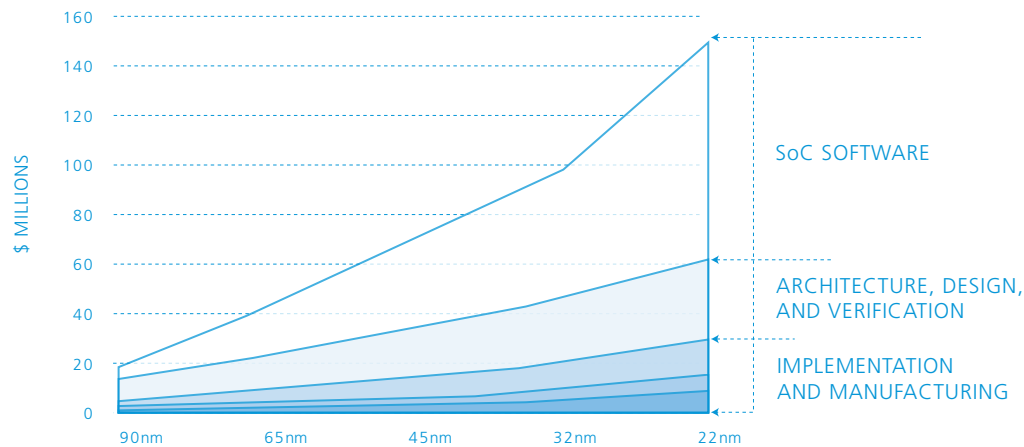


Figure 1 – SoC development costs have soared from \$20 million at 90nm to nearly \$100 million at 32nm. Software is the fastest growing part of the cost. [source: IBS]

To be profitable and to amortize this \$100 million investment, a semiconductor company may need to ship 80 million units. Considering that Apple had shipped a little more than 40 million units of the iPhone by the end of 2009, 80 million units is a huge target.

In the past, semiconductor companies would design six to ten ICs, realizing that perhaps only one or two of them would catch fire and generate revenues to pay for all the rest. At advanced process nodes that is no longer feasible. Every semiconductor design is critical and must be applicable to a broader market. Just one development project that goes nowhere can easily sink a company.

The EDA industry to date has only served the needs of creators. It has almost completely ignored integrators, who need a different set of tools and capabilities.

A few large semiconductor companies will continue to follow Moore's Law and design the fastest, most complex, and smallest ICs. These innovators are design *creators*. While they will provide a crucial role in the industry, only a handful of such companies can exist.

Clearly there must be a way to be successful at lower volumes or with less advanced silicon. Consumer demand is setting the stage for new kinds of connected devices we haven't even imagined yet. If electronic design is only available to a handful of creators who can only make money by shipping 80 million units, few of these devices will be built and the diversity that consumers want will not materialize.

Fortunately, there is another way. Some innovators will redefine themselves as *integrators*. They will integrate at the silicon, SoC, and system levels. They will make heavy use of externally designed silicon and software intellectual property, they will tend to stay at mature process nodes, and they will invest heavily in embedded software development. They will become application-focused platform providers, not "chip" providers. These integrators must reach an elusive profitability target and get there through rapid integration without sacrificing quality or schedule.

The integration task involves a different set of challenges than creation. Integrators need to be able to locate IP, evaluate it, and source it more effectively than they ever have before. They need to integrate IP into their platform hardware, using whatever configurability is required. They need to verify and test their platforms and SoCs. Integrators also need to integrate and verify embedded software. Furthermore, since all real-world consumer applications involve both analog and digital circuitry, integrators will need to build and verify "mixed-signal" platforms.

Both creators and integrators are ultimately concerned about cost, but they look at the problem in different ways. Creators are most concerned about delivering silicon with the best performance, the lowest power, and the smallest die size. Integrators want hardware that is "good enough" to serve application needs, but their biggest concerns are quality, cost, and time to market.

The EDA industry to date has only served the needs of creators. It has almost completely ignored *integrators*, who need a different set of tools and capabilities. Where, for example, are the tools for silicon IP evaluation, quality assurance, and snap-in integration? Why is so little silicon IP "integration-ready?" And what is being done to ease the verification of analog/digital or hardware/software interfaces?

When all you have is a hammer, everything looks like a nail. A growing number of integrators are demanding something better.

FROM PRODUCTIVITY TO PROFITABILITY

The number one concern of the creator is productivity. There has been much discussion about a growing “productivity gap” in IC design as process nodes shrink and complexity grows. Simply put, the productivity gap is the difference between silicon capacity and engineering output—the difference between what could be, and what actually is.

Closing the productivity gap is far from a solved problem, and it is one that EDA providers can and should continue to address. Effective productivity management brings innovative and differentiated capabilities to design, verification, and implementation.

At the *design* level, productivity will be enhanced by moving to a higher level of abstraction and by increasing design reuse. At the *verification* level, productivity will be improved with a metric-driven approach encompassing simulation, formal verification, and emulation. At the *physical implementation* level, EDA technology must keep up with growing digital design complexity at advanced nodes, and support manufacturing-aware, mixed digital, and analog co-design.

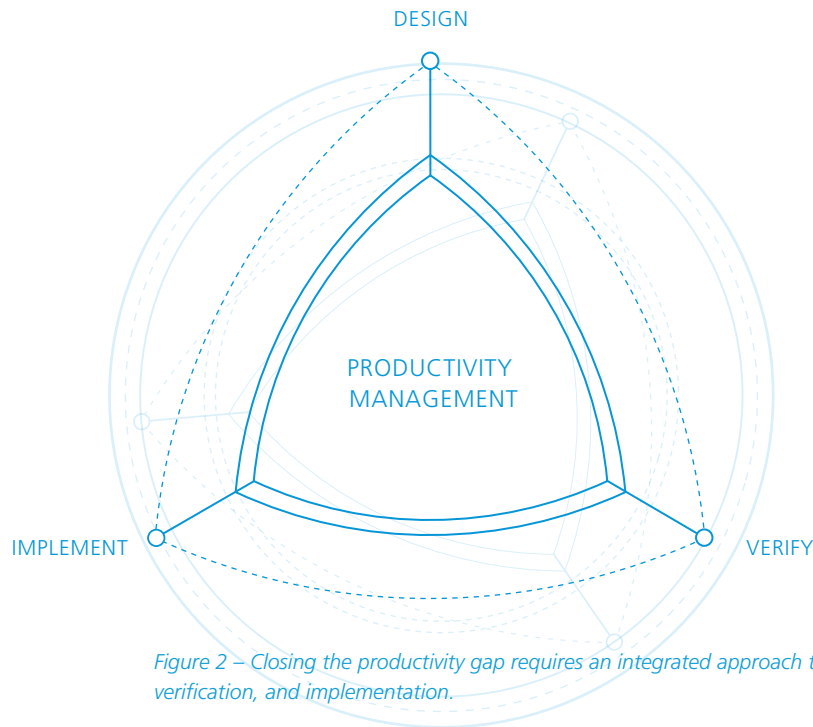


Figure 2 – Closing the productivity gap requires an integrated approach to design, verification, and implementation.

Design, verification, and implementation must be closely integrated. All too often these tasks are performed in isolation with limited feedback among engineering teams.

While the EDA community can and should continue to improve productivity, what the integrator most cares about is profitability. There is a growing “profitability gap” in the electronics industry that has rarely been discussed. It is the gap between business goals and design cost, semiconductor unit cost, and time to market (or “delay cost”). In short, it is the difference between what you can make and what you can make money on.

To close the profitability gap, companies must control hardware/software development costs and lower the costs of packaging, manufacturing, and test. Beyond controlling costs, integrators must increase revenue. This can be done by meeting tight time-to-market

windows, and by designing the right product for the right market at the right time, avoiding “feature overshoot” that provides more than consumers are willing to pay for.

The EDA industry must help close the profitability gap. Closing that gap involves three steps:

- **Create.** Whenever or wherever hardware or software IP is developed—be it for external or internal use—it must be created with integration in mind. It must come with the right documentation, be fully verified, and be configurable for the end application.
- **Integrate.** Design teams must be able to rapidly integrate IP into platforms, and then verify the complete SoC or platform, including interfaces between analog and digital blocks as well as hardware and software.
- **Optimize.** Design teams must be able to reduce die and package costs while ensuring quality, lowering power consumption, and trimming test costs through automation. They must reduce the number of late-stage iterations.

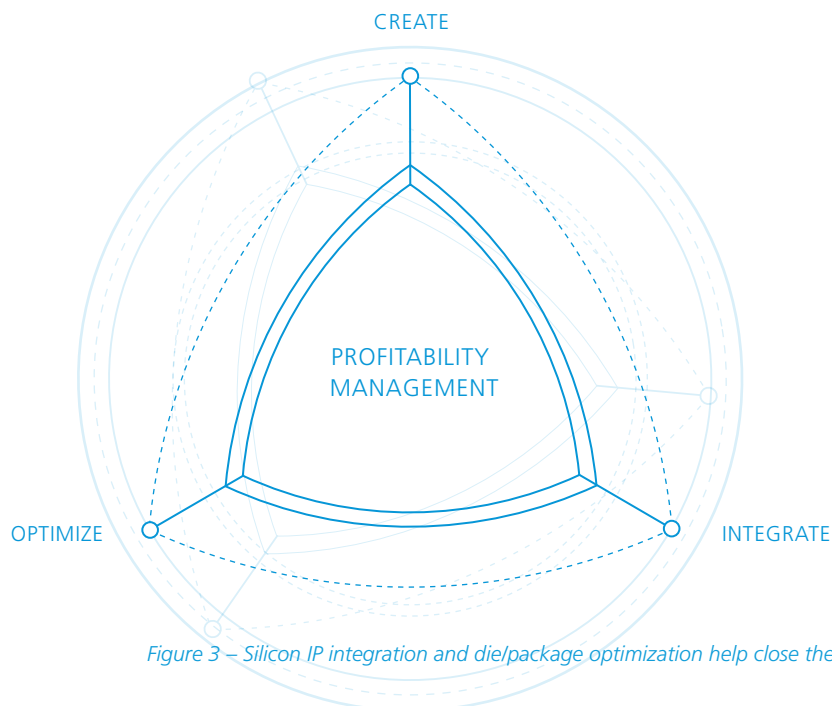


Figure 3 – Silicon IP integration and die/package optimization help close the profitability gap.

Finally, a collaborative ecosystem including semiconductor design companies, IP providers, service providers, foundries, EDA vendors, and assembly/packaging houses is needed, along with the enablement of open, standards-based solutions. Without such an ecosystem, the electronics industry will collapse back into a handful of large, vertically-integrated companies.

While the EDA community can and should continue to improve productivity, what the integrator most cares about is profitability. There is a growing “profitability gap” in the electronics industry that has rarely been discussed.

With EDA360, users start with an understanding of the software applications that will run on a given hardware/software platform, define system requirements, and then work their way down to hardware and software IP creation and integration.

TOWARDS THE NEXT GENERATION OF EDA

EDA360 provides an expanded, 360-degree vision of EDA that serves integrators as well as creators. EDA360 helps close the profitability gap through integration-ready IP creation, IP integration, and optimization. Given that embedded software can take up half the cost of SoC development, EDA360 also supports hardware/software integration and verification. It thus expands the scope of EDA well beyond its original boundaries.

While traditional EDA focuses on engineering teams only, EDA360 will provide capabilities for project and business management. It will reach across a customer's entire global organization—engineers, project managers, and corporate leaders—to enable profitability and competitiveness in these challenging times.

With EDA360, users start with an understanding of the software applications that will run on a given hardware/software platform, define system requirements, and then work their way down to hardware and software IP creation and integration. EDA360 supports three important capabilities:

- **System Realization** is the development of a complete hardware/software platform that will provide all necessary support for end-user applications. The platform includes one or more SoCs developed through SoC Realization, and adds an embedded software infrastructure that typically includes an OS, middleware, and reference applications. System Realization is driven by the applications that will run on the completed system. To be successful, it requires enterprise-level program management.
- **SoC Realization** is the completion of an individual SoC (or alternative packaging choice, such as 3D IC). Along with the integration of silicon IP developed through Silicon Realization, SoC Realization includes “bare-metal software” such as drivers and diagnostics. In the EDA360 vision, an Open Integration Platform facilitates IP integration into SoCs.
- **Silicon Realization** represents everything it takes to get a design into silicon. The result could be an analog or digital IP block for an SoC, an IP subsystem, or a complete IC without embedded software. Silicon Realization increasingly involves the creation and integration of large and complex digital, analog, and mixed-signal IP blocks. It goes well beyond conventional “mixed-signal” designs that integrate a few small analog blocks into a digital SoC.

IP creation, reuse, and sourcing are at the core of the EDA360 vision. IP integration takes place during all three steps listed above, with software IP coming into play during SoC Realization and System Realization. Integration is where “what is needed” from the top meets “what is possible” from the bottom.

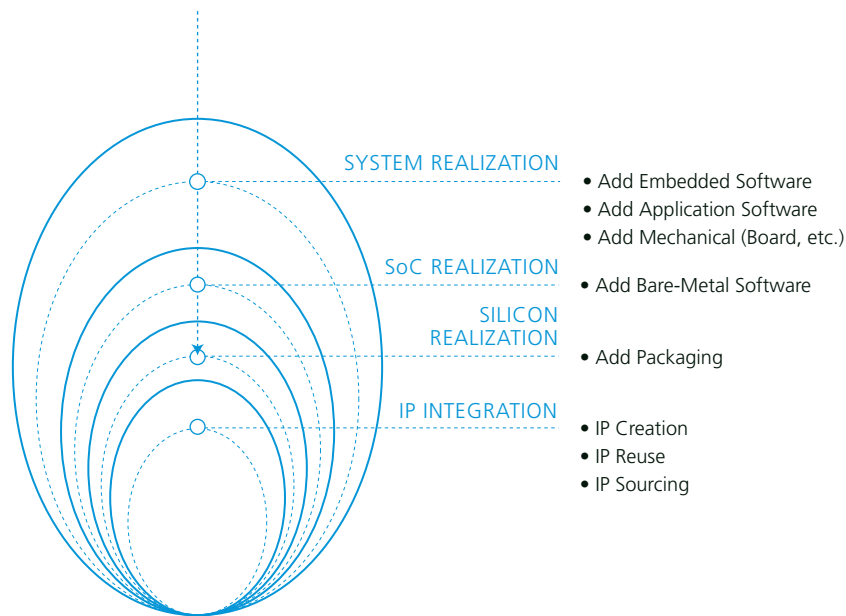


Figure 4 – EDA360 extends down from System Realization to SoC and Silicon Realization.

CONCLUSION

EDA360 will close the profitability gap through creation, integration, and optimization, and it will provide badly needed support for both creators and integrators. As such, it will ensure the continuance of a vibrant, pioneering electronics industry with thousands of players. That industry will not only change the way we work and play—it will provide the solutions we need for energy conservation, the environment, health care, education, and transportation.

CHAPTER 2: APPLICATION-DRIVEN SYSTEM REALIZATION

To gain more insight into EDA360 and the world it will shape, it is best to start with a discussion of System Realization. Today's most commercially successful systems are application-driven, which means their design has to be done in that context. This requires a platform that will allow end-user applications to meet their design intent and promise, positively impacting both profitability and engineering productivity.

Electronics products today are all about the applications. As such, we are seeing a new approach to systems design in which companies begin projects with an understanding of the needs of the applications that will run on the end system. Then, they use that understanding to develop an optimized platform to support the applications. In addition to the hardware, this platform includes the bare metal (hardware-dependent) and operating system layers of the software.

Traditionally, in contrast, most systems are developed from the bottom up, starting with the hardware. The OS already exists or is developed with a very generic understanding of the underlying hardware, and the applications are written within the limitations of the pre-determined hardware and software infrastructure capabilities.

There has been much talk in the electronic design world about what a system comprises. In this discussion, a “realized” system is a hardware/software platform ready for the application(s). System Realization must comprehend all levels of the software stack, from the bare-metal software and OS to the libraries and applications.

Electronics products today are all about the applications. As such, we are seeing a new approach to systems design in which companies begin projects with an understanding of the needs of the applications that will run on the end system.

The *platform* concept is central to System Realization. In the semiconductor industry, integrators build configurable platforms constructed from libraries of components such as intellectual property blocks to meet the stringent demands of consumerization—interoperability, fast time to market, and low cost. In SoC Realization as described in Chapter 3, these IP blocks are not just traditional silicon IP. They are part of an optimized IP stack—from bare-metal software to the physical layer—that allows visibility into and control of the hardware from the OS and the application.

A well-known platform example in the semiconductor industry is the NXP Nexperia, which is used for connected multimedia devices. In addition to programmable SoCs and companion ICs, Nexperia comes with reference designs, system software, and development tools. No less important is NXP’s partnership program, which builds an ecosystem around Nexperia solutions.

A DISJOINTED APPROACH TO SYSTEM DEVELOPMENT

The traditional approach to system development is disaggregated. In most cases, hardware design is disconnected from software infrastructure development. To create hardware, system and SoC design teams create or buy silicon IP. This may be “hard” IP with fixed physical layouts, or “soft” IP that can be converted to a gate-level representation with a logic synthesis tool. Although more flexible than hard IP, soft IP is typically at the register-transfer level (RTL), a relatively low level of abstraction that assumes a fixed micro-architecture. Designers continue with implementation and verification until the hardware is designed and silicon IP integration is complete.

While the operating system is generally pre-selected, the application development is largely abstracted from the hardware. In the absence of a virtual prototype or emulator—technologies not yet widely used by software developers—applications cannot be completed until the hardware is physically available. It then becomes a frenzied effort on a short time schedule, carried out by programmers with little or no control of, or visibility into, the hardware resources. Hardware/software integration and debugging occurs late in the design cycle, causing schedule delays, cost overruns, and quality problems.

The end-user applications are then developed, most likely by a third party who has little familiarity with the underlying hardware or software. Applications must conform to any limitations imposed by the hardware or software. Unless it is over-designed, the underlying platform will probably support a narrow range of application requirements, limiting the creativity of applications developers.

Application, software, and hardware development are done by different teams using their own tools and IP. Teams rarely communicate and are usually located in disparate geographies. The result is a deep chasm between each of the three steps—applications, software, and hardware.

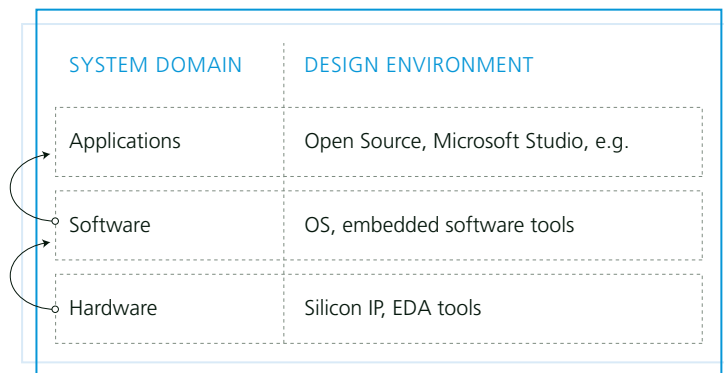


Figure 1 – In traditional system development, hardware design comes first and software is appended later.

Aside from time-to-market delays, there is an inherent problem with this disaggregated approach. Any glitches that potentially involve more than one of the three domains listed above are very difficult to resolve. If an application crashes, is the problem in the application itself, the infrastructure software, or the hardware? If an infrastructure software program experiences a memory error, is the bug in the software or in a hardware register?

Many design teams lack the tools or knowledge to track down such problems quickly. Instead, it's often a trial-and-error effort with a lot of iteration—and sometimes no resolution.

If hardware is designed first, and software is appended later, it is difficult to optimize at the system level. Today, for example, many systems must be optimized for power consumption. Advanced EDA design flows support power optimization techniques such as power shutoff or multiple voltage levels. But if software is not optimized to take advantage of such features, they won't be very effective. For effective low-power optimization, software and hardware should be considered concurrently.

To support the range of applications consumers have come to expect, we must turn the hardware-first paradigm on its head and move toward an application-driven System Realization approach.

MOVING TOWARD APPLICATION-DRIVEN DESIGN

With application-driven System Realization, you don't start by building hardware—you start by envisioning the applications you want to run. Then you need a hardware platform along with system software, including drivers, an OS, and middleware. How do you obtain the hardware/software platform that's right for your application? As of today, there are two different approaches to application-driven design—hardware-defined and hardware-independent.

In a hardware-defined approach, the hardware is pre-designed and tightly integrated with all system software below the application level. A software development kit (SDK) provides a single view of the system, along with everything developers need to create applications that run effectively on the platform. The Apple iPhone uses this approach.

With a hardware-defined methodology, application developers have a stable, predictable platform upon which to build a variety of applications. They can develop applications that are also stable and predictable, and build and test them quickly, without any special knowledge of the underlying software and hardware. While these are tremendous advantages, applications must be written in conformance with the platform’s capabilities. The hardware cannot be reconfigured or changed. From a hardware standpoint, it’s a “closed” system.

In a hardware-independent approach, no underlying hardware is defined or specified. This is the approach Google has taken with Android, a software stack for mobile devices that includes an operating system, middleware, and key applications. Android is a free, open-source offering that comes with an SDK that provides the tools and application programming interfaces (APIs) needed for developing applications. Android provides an “open” system.

With the hardware-independent approach to application-driven design, you can source or build a platform optimized for your type of application—not too much capability, not too little, but just right. In contrast, a hardware-defined platform cannot be reconfigured or changed, and it is unlikely to be fully optimized for the application you want to run.

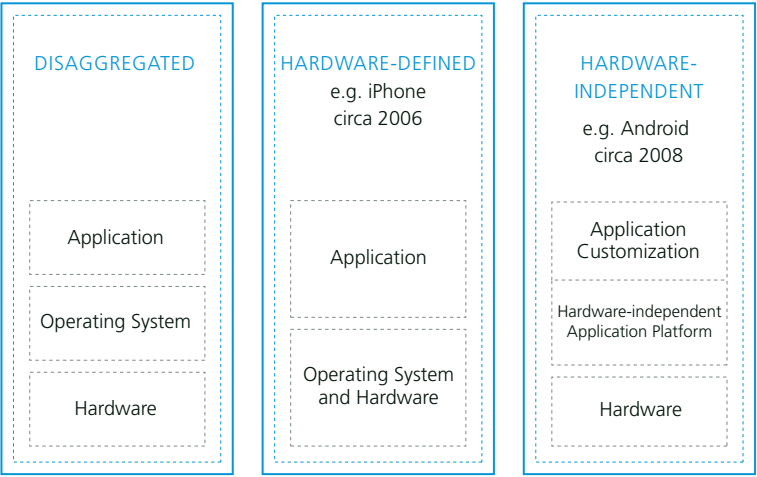


Figure 2 – Hardware-independent applications development makes it possible to create the right hardware for a given application.

This does not mean that you would build or source custom hardware for every application. Rather, with a configurable hardware/software platform, the software application can reconfigure the hardware in a way that suits the application. Suppose you have a web application that includes video. The application will want to open the connection between the video graphics driver and the memory as widely as possible, perhaps closing down other functions not needed for a web refresh. Later on, when it’s time to do some computation, the hardware will be configured differently. The application, rather than the CPU, is prioritizing resources.

Application-driven System Realization points to a 180-degree turnaround from the disaggregated, hardware-first development model. Instead of system capabilities driving the applications, the applications are driving the system requirements. By developing a solution that is “just right” and not over-designed, integrators can satisfy their primary concerns—cost, time to market, and quality—and close the profitability gap.

To support the range of applications consumers have come to expect, we must turn the hardware-first paradigm on its head and move toward an application-driven System Realization approach.

HOW EDA NEEDS TO RESPOND

To support application-driven System Realization, EDA360 must help provide new capabilities. Five such capabilities are as follows:

- Early software development
- Application-driven system integration
- Application-driven system verification
- Driver development kits
- Application development kits

It's important to note that today no single vendor can provide all the solutions that are needed for hardware/software creation, integration, and verification. That is especially true at the System Realization level, where both embedded software and hardware expertise are required. Ecosystem collaboration will be essential for success, as will open standards that provide choice for customers. EDA vendors will need to partner with embedded software companies, IP providers, and customers to provide many of the capabilities described below.

EARLY SOFTWARE DEVELOPMENT

From an applications perspective, you simply want to get your applications up and running as soon as possible on a hardware/software platform that meets user requirements. You certainly don't want a long, slow, serial process in which application development isn't started until the hardware is complete. Ideally, software development will start as soon as processors are identified and there's a rough idea about the overall architecture.

Virtual prototypes today let software developers create and debug software using fast processor models and transaction-level models of peripheral hardware, well before any hardware is actually built. This capability reduces time-to-market and cuts development costs. When timing accuracy is required, however, hardware emulation tools are needed. Emulators can find timing-dependent bugs that a virtual prototype would miss, and emulation is now available with dynamic power analysis.

Finally, better software development support is needed for “multi-core” SoCs with multiple processors. This is especially true with heterogeneous multi-core SoCs that have different types of processors and different operating systems. Today, developers do a lot of manual partitioning and trial-and-error work. Better solutions are required, and they will require a collaborative effort that includes EDA vendors, embedded software providers, and the academic and industrial research communities.

APPLICATION-DRIVEN SYSTEM INTEGRATION

SoCs are assembled from libraries of analog, digital, and mixed-signal IP blocks. These IP blocks are sometimes created internally but are more often purchased from third-party companies or acquired from other design groups. Putting IP together effectively and efficiently is a significant challenge that is not well served by the EDA industry today. As a result, some companies report that IP integration can cost more than twice as much as it costs to buy the IP in the first place.

System Realization can be viewed as an IP integration challenge, although at this level of abstraction the IP might be an entire SoC, an operating system, or a reference application. Hardware/software integration is taxing at the system level, especially when hardware is designed first and the software is appended later with limited control or visibility into hardware. The hardware-software interface is one of the likeliest places for bugs to occur, and when they do, it's very difficult to determine whether they came from the hardware, firmware, software, or application.

An application-driven approach eases hardware/software integration challenges. Here, you can start software development as soon as a few basic decisions (such as a processor) have been made, and then abstract the hardware requirements from the software. What-if analyses are extremely helpful at this stage. Chip-planning tools that offer rapid, early estimations of power, die size, yield, and cost are now available. Over time, "system planning" tools that consider both software and hardware will emerge.

No single vendor can provide all the solutions that are needed for hardware/software creation, integration, and verification. That is especially true at the System Realization level, where both embedded software and hardware expertise are required. Ecosystem collaboration will be essential for success.

As integration proceeds, an automated engineering change order (ECO) capability is needed at every level of abstraction. Changes are inevitable, and they cannot disrupt the design cycle. It should not be necessary to tear up a system or re-synthesize an entire IP block because of one small change. The harder it is to change a hardware/software platform, the greater the need for a perfectly specified platform.

Automated ECO tools are available today. These tools formally verify changes, and allow incremental design changes that minimize disruption. This capability is still focused on hardware, but has recently been extended to high-level synthesis.

APPLICATION-DRIVEN HARDWARE VERIFICATION

Hardware design teams know that verification consumes most of the hardware development time and cost. Software verification and debugging are equally time-consuming. At the System Realization level, integrators must verify that the entire system meets all requirements and functions as expected.

System Realization requires hierarchical verification. That means you should be able to verify IP once, check it off, and then only verify inter-block interfaces at the next level up in abstraction. By the time you get to System Realization, only the interfaces between major system components should remain. In the real world, unfortunately, a lot of re-verification

occurs, and verification across analog/digital or hardware/software boundaries is extremely complex. To make things more complicated, system-level verification teams must run extremely long test sequences to find rare “corner case” bugs.

The metric-driven verification approach advocated by Cadence provides a huge advantage for hardware developers today. It allows design teams to start with an executable verification plan and track coverage metrics that show how complete the verification process is. Bringing this concept into the embedded software world, where a formalized verification methodology is lacking, would greatly benefit System Realization. Collaboration between IBM and Cadence has shown that a single verification plan can control and track both hardware and software verification.

Part of EDA360 is providing new tools that offer a “dashboard” to help companies manage system development projects, and provide metrics that make sense in hardware and software engineering environments.

Mixed-signal verification at the system and SoC levels is extremely challenging. Analog simulators are far too slow for system-level verification, and even with behavioral modeling techniques, the interface between analog and digital circuitry is very complex. But digital simulators—which are orders of magnitude faster than analog simulators—can, in fact, simulate analog signals if they take advantage of language constructs now available in standard analog modeling languages.

With the focus on applications, the nature of verification changes. It is no longer enough to just run stimulus and look at waveforms or source code. To verify a video application at the System Realization level, you want to see the video. To verify a music application, you want to hear the music. In other words, you want to verify applications in the context of the full system. It is possible to run portions of live applications today using virtual prototypes or emulation tools.

DRIVER DEVELOPMENT KITS

Today, software driver development for custom hardware is a manual effort. If you start with an application and create or integrate hardware, developing the necessary drivers can be a big obstacle. And with many hardware options and port configurations, you may end up with any number of different drivers for the same hardware platform.

A driver development kit can help. Imagine, for example, a Linux driver development environment that includes templates that make it easy to write drivers for embedded Linux, showing software developers how to specify ports and handle interrupts.

APPLICATION DEVELOPMENT KITS

Apple provides an efficient software development kit for application developers. The kit abstracts away details about the underlying hardware and software, providing only what developers need to know to make applications work on the platform. The drawback is that the hardware cannot be optimized or customized. Android SDKs understand Android, but obviously not hardware that Google doesn't provide.

There is a place for application-driven SDKs that are not based on fixed hardware or software platforms, but still understand basic functions that are required for a particular class of applications. For example, a kit for mobile applications would understand call handling and routing, as well as requirements for analog or radio frequency (RF) support.

MANAGING THE SYSTEM REALIZATION PROCESS

The development environment that enables application-specific System Realization is itself a "system" that needs ongoing management and guidance. Much like enterprise software provided for businesses, enterprise-level management is needed for cost, schedule, and personnel resource management aspects of systems development. These tools need to manage geographically-dispersed teams as well as Silicon Realization projects that reach across multiple companies.

Enterprise resource planning (ERP) tools provide some of these capabilities, but they neither have access to the underlying data nor an understanding of the challenges and requirements of electronic system development. A tool unaware of "system integration" cannot keep you on schedule for system integration. A tool with no understanding of "verification" cannot help you track and monitor the verification process.

Thus, part of EDA360 is providing new tools that offer a "dashboard" to help companies manage system development projects, and provide metrics that make sense in hardware and software engineering environments. The metric-driven verification capability available now points the way to what needs to be done. Today, design and verification managers can create an executable verification plan that identifies key project metrics, executes simulation engines, and tracks coverage metrics. They can then review reports and charts that will help them manage a "plan to closure" verification process, and determine when verification is done. As a result, verification resources are used effectively and overall costs are reduced, helping close the profitability gap.

CONCLUSION

The traditional approach in the semiconductor industry is to build the hardware, append software later (if we really have to!) and let somebody else worry about the applications running on pre-built hardware. But in a competitive consumer environment where differentiation comes from the latest creative applications, and profitability is hard to attain, this conventional approach is running out of steam. With an application-driven System Realization approach, developers can start by envisioning the application. They can then design at the system level as far as possible, work down to the software, and finally build or buy the hardware.

The application-driven approach will help close the profitability gap by addressing cost, time to market, and quality. But it's a completely new way of looking at design, and it places new demands on system integrators and EDA providers. EDA360 is gearing up to meet those demands.

CHAPTER 3: SOFTWARE-AWARE SoC REALIZATION

While System Realization produces a complete hardware/software platform ready for applications deployment, SoC Realization ensures the successful development of a single SoC to meet system needs. Typically, SoCs are considered to be “done” when the silicon is completed. In the EDA360 view, however, SoC Realization is not complete without software device drivers for each hardware subsystem. We believe these drivers should be developed with the SoC rather than tacked on later—and that leads to a completely new view of how silicon IP should be provided.

Instead of thinking of IP as isolated “blocks,” we propose an IP stack that includes “bare-metal software” as well as hardware IP. Bare-metal software refers to everything below the OS layer, and the most prominent feature of bare-metal software is device drivers. The IP stack depicted below also includes verification IP (VIP) that validates IP functionality and integration. The stack may include hard macros with fixed layouts along with synthesizable IP at the register-transfer level or the transaction-level modeling (TLM) level. It also includes design constraints.

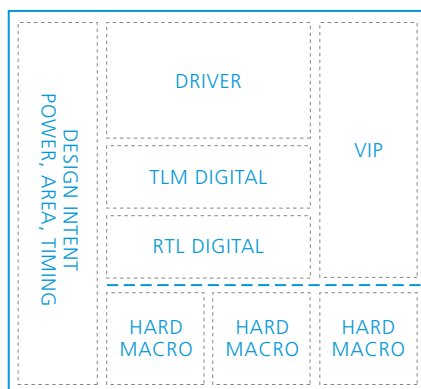


Figure 1 – An IP stack includes synthesizable RTL or TLM IP, verification IP, and hard macros, in addition to driver software and design constraints.

Driver software allows the OS and application to manage hardware resources. As such, all electronic systems must eventually incorporate device drivers. So why bundle them with silicon IP and include them as part of SoC Realization?

Consider how drivers are typically developed. In the conventional SoC design methodology, hardware is built first and drivers are written later, usually by someone unaware of the details (and thus the differentiating capabilities) of the hardware. This leaves two bad choices. One is an expensive, custom driver development effort by someone who probably has little hardware knowledge—or if it’s a hardware person, little or no OS knowledge. The other choice is to purchase a generic driver that will have only a general understanding of how a given type of hardware should appear to the OS, and won’t take advantage of all of the capabilities of your hardware.

In either case, the applications are disconnected from the underlying hardware system. The result is a weak link between the application and the hardware. You can eventually build a system that will more or less work—but it will probably be sub-optimal, over-designed or under-designed, delayed, and probably too expensive to attain profitability.

If drivers are part of an IP stack, however, they will present the full range of underlying hardware capabilities to the operating system. This makes it easier for the OS to directly control or configure hardware resources to meet the needs of the application. For example, a video application can directly and immediately call upon the bandwidth it needs to do a fast web refresh. It can then configure the hardware differently to run some computation. This capability is diminished if drivers don't fully comprehend the hardware.

SoC Realization is not complete without software device drivers for each hardware subsystem. We believe these drivers should be developed with the SoC rather than tacked on later—and that leads to a completely new view of how silicon IP should be provided.

It is also important to include design constraints with the IP stack. The most common constraints are those for power, timing, and area. These constraints come from various sources, including Common Power Format (CPF) files for power, timing scripts for logic synthesis, and physical layout constraints for analog blocks. Constraints are an expression of design intent, and they are—or should be—developed very early in the design cycle. To preserve design intent, constraints must be comprehended throughout the design, verification, and implementation steps that are necessary to close the productivity gap.

So why include constraints with the IP stack? Because without a thorough understanding of design constraints, the integration of IP into an SoC will be difficult and error-prone. Bugs are more likely to appear in the interfaces between IP subsystems, overall system functionality may be compromised or faulty, and verification will be more costly. A well-specified, thorough set of design constraints is thus an important aspect of “integration-ready” IP. Constraints should be provided for all parts of the stack, including analog hard macros, synthesizable digital IP, and driver software.

A BROADER DEFINITION OF SoCS

From a silicon standpoint, an SoC is a configurable or programmable IC that includes at least one processor, some custom logic, and a memory controller. An increasing number of SoCs are multi-core ICs that have multiple processors of the same or different types, such as CPUs, GPUs, DSPs, and specialized hardware accelerators. SoCs may also have on-chip memory, peripherals, and protocol interfaces. All of these components are organized into subsystems—a CPU subsystem, a memory subsystem, an I/O subsystem, and so on.

Given the increasing complexity of today's SoCs, it is vital to ensure efficient communication among these subsystems. If the communication architecture is inadequate, connecting subsystems together will be painful at the implementation stage. Thus it is essential to comprehend that communication network at the architectural phase of SoC Realization. A newer on-chip communication approach called “network on chip” (NoC) is promising increased flexibility for multi-core SoCs.

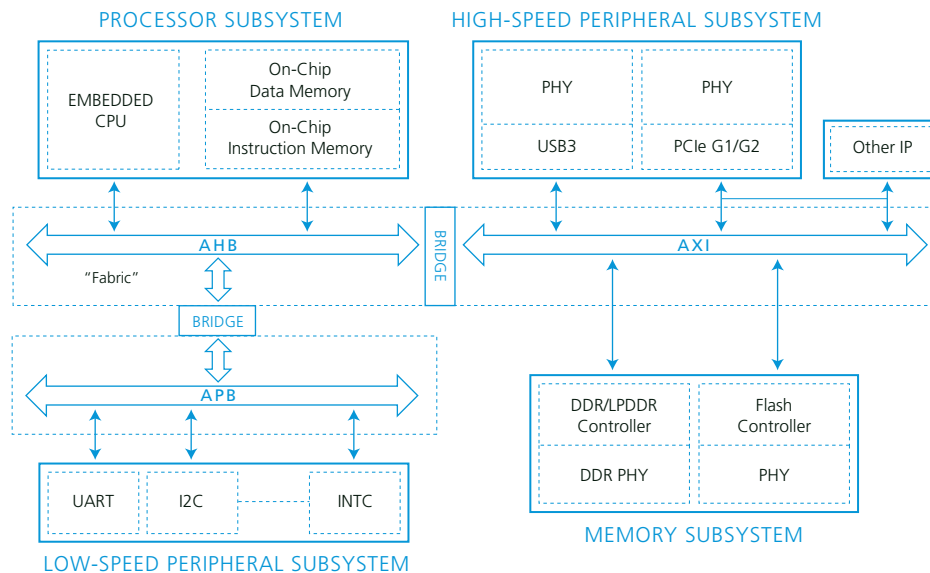


Figure 2 – SoCs are organized into subsystems. One of the subsystems provides a communications fabric for the others.

So far we've used a conventional definition of SoCs. With software-defined SoC Realization, however, drivers are provided with each subsystem. The drivers allow the OS to directly take advantage of specialized hardware features in each subsystem, including the on-chip communications fabric.

SoC Realization is most efficient when multiple IP stacks are assembled into IP subsystems. These, in turn, support major hardware subsystems, such as I/Os, CPUs, or memory. IP must also be optimized for integration. Design constraints are an important part of that optimization.

INSIDE THE IP STACK

Let's take a closer look at the IP stack. Conventionally, most digital silicon IP includes a controller layer and a physical layer (PHY). For standards-based IP, such as PCI Express or USB, the controller is usually "soft" or synthesizable IP, while the PHY is typically "hard," meaning that synthesis and layout have been completed. CPU IP, however, is usually entirely synthesizable.

Soft IP is typically built at the register-transfer level today. There's a growing move to a higher level of abstraction, transaction-level modeling, which will allow much faster design and verification and a choice of micro-architectures. Transaction-level models can also be used to build virtual prototypes for early software verification. With the recent availability of SystemC® high-level synthesis with a high quality of results, the necessary tools for a migration to TLM are already in place.

But TLM-based digital IP is just one part of the story. Verification IP has become increasingly important and should be part of the IP stack. VIP provides the complex testbenches that are needed to verify IP configuration and integration. In addition to testbenches, VIP is available now with protocol compliance management, support for assertions, transaction-based acceleration, and rate adapters for emulation.

Since nearly all SoCs are mixed-signal, an IP stack will likely include some analog hard macros. Chip-level, mixed-signal verification is one of the toughest challenges faced by SoC and system designers today. One newly available solution makes it possible to simulate analog signals inside a digital simulation environment, while taking advantage of the metric-driven approach available for digital verification.

What's really new about the IP stack discussed here, however, is the inclusion of device drivers and design constraints. When the IP creator provides the drivers, it's no longer necessary for someone with little or no hardware knowledge to build them after the fact. When the IP provider includes constraints, the SoC integrator can assemble IP stacks and subsystems with confidence.

With this new IP approach, does one architect and implement an SoC differently? The answer is an emphatic "yes" and is found in a new concept called the Open Integration Platform.

OPEN INTEGRATION PLATFORM

SoC integration involves three key steps:

Analyze the architecture. Based on the application requirements, system designers perform what-if analyses (business and technical), develop an overall plan for the SoC, and define the constituent elements such as processors, I/Os, and memory. They also define the bare-metal software that allows the OS layer to control the hardware. Chip planning tools are currently available to help with some of these tasks by allowing rapid, what-if analyses for power, area, and cost.

When the IP creator provides the drivers, it's no longer necessary for someone with little or no hardware knowledge to build them after the fact.

Develop or source integration-optimized IP. Whether it comes from internal sources or third parties, designers must be able to locate the IP that's best for the application from the many sources available. This requires a catalog or library of IP that allows technical and cost comparisons, as well as tools that permit what-if analysis for power, performance, and cost. In some cases, designers will decide to create integration-optimized IP. Some of the requirements of integration optimization include:

- **Functionality is well-defined and documented**
- **Source, synthesis, and implementation packages are provided for integration**
- **Silicon-validated IP is available with characterization data**
- **IP meets documented quality guidelines**
- **IP blocks are parameterized**
- **IP stacks come with design constraints**
- **Test platform for IP comes with a complete verification environment for hardware and drivers**
- **Deliverables include all necessary scripts, guidelines, checklists, and documentation**

Integrate IP to realize the SoC. With individual blocks of IP, integration is a daunting task. When IP is organized into pre-verified, integration-optimized subsystems, it becomes much easier. Designers need to be able to “snap together” subsystems. The *design*, *verify*, and *implement* steps needed to close the productivity gap described in Chapter 1 apply here, and are evolving to meet the needs of the new IP integration paradigm. Productivity continues to be an important part of the SoC Realization process and also serves to close the profitability gap.

An Open Integration Platform provides the framework within which to carry out these steps. It is driven by an Integration Design Environment that is analogous to the IDE (Integrated Development Environment) used by software developers. It provides the comprehensive set of capabilities to develop today’s SoCs, maximizing both productivity and profitability. It is based on open standards like the OpenAccess database, the Open Verification Methodology (OVM), the IP-XACT format for IP descriptions, and the SystemC TLM 2.0 modeling standard.

Working from this SoC-IDE (Integration Design Environment), designers will take in customer-specific and application-specific requirements, and bring in IP subsystems from internal or external sources. They may make use of third-party customization services to provide expertise needed for short periods of time—for example, to bring in some analog expertise to resolve any questions around mixed-signal subsystems. Integrators can design, verify, and implement any additional IP that is needed, and then assemble and verify the completed SoC, providing all the information necessary for fabrication or handoff.

The SoC-IDE provides a “dashboard” for underlying design, verification, and implementation tools. It thus takes advantage of existing capabilities such as metric-driven verification, low-power design, mixed-signal implementation and verification, and hardware/software integration. These capabilities will continue to evolve to meet the needs of SoC integrators.

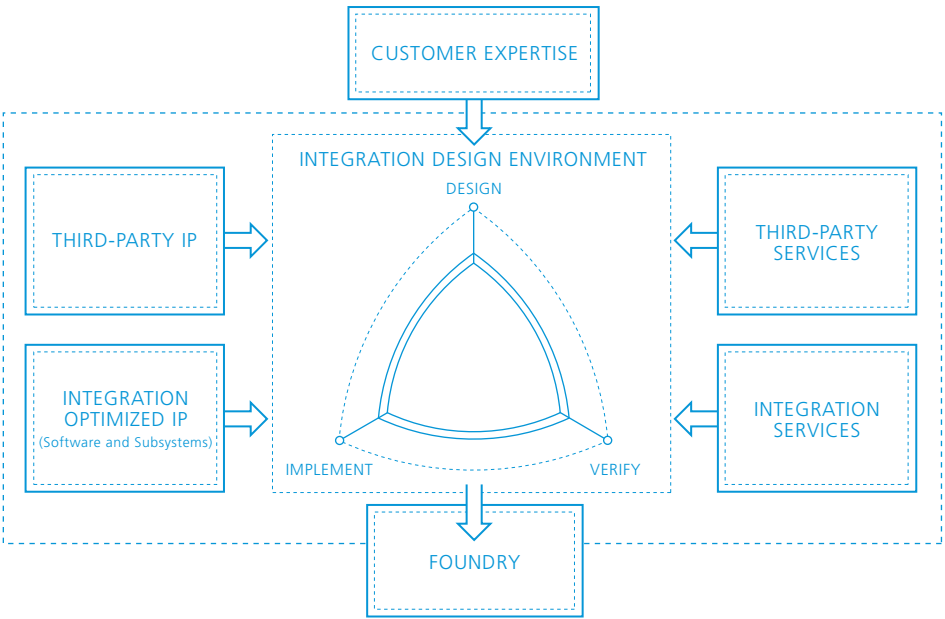


Figure 3 – An Open Integration Platform makes it possible to produce an integrated SoC from IP subsystems.

SoC Realization as described here calls on designers to create integration-optimized IP when needed, to integrate both internally created and sourced IP, and to then verify and optimize the SoC to meet system requirements. Thus, the steps that close the profitability gap—*create, integrate, optimize*—are crucial to SoC Realization as well.

CONCLUSION

The EDA360 view of SoC Realization calls for an expanded definition of SoCs that includes the bare-metal software, along with a more comprehensive way of looking at IP as a “stack”—from the physical layer and the protocol (control) layer to the drivers along with associated verification IP and design constraints. By including driver software, a critical weak link is repaired, allowing the applications to work through the OS to call upon the precise hardware resources it needs.

To fully support this vision of SoC Realization, new technology is required. One step is to make it easier for IP creators to build drivers. Another step is to facilitate thorough, metric-driven IP verification to the point where re-verification of IP is unnecessary. And EDA360 will need to provide an Open Integration Platform, with its ability to run what-if analysis to determine the best possible architecture and integration, develop an initial SoC architecture, and quickly integrate verified IP into verified SoCs.

The type of IP we’ve described here can only come about through a collaborative ecosystem. EDA companies, embedded software and OS vendors, IP providers, foundries, and end-user companies must all work together to expand the definition of IP to bare-metal software and complete subsystems.

The approach outlined in this chapter will open the door to profitable, application-driven System Realization. Instead of spending time and money to have a specialist develop custom drivers, or accepting the limitations of a “generic” driver, SoC integrators will have driver software that presents the capabilities of the underlying hardware. Instead of flying blind and running into problems with integration, SoC developers will follow a constraint-driven integration methodology. The approach described here helps integrators meet the goals of cost, time-to-market, and quality, therefore closing the profitability gap.

With individual blocks of IP, integration is a daunting task. When IP is organized into pre-verified, integration-optimized subsystems, it becomes much easier.

CHAPTER 4: EDA360 ENABLES SILICON REALIZATION

EDA has always been concerned with Silicon Realization. In fact, that's been nearly the entire focus of the EDA industry since its inception. From an EDA360 point of view, Silicon Realization is no longer the starting point for system development—yet it remains as essential as ever, and it's becoming much more challenging.

Silicon Realization represents everything it takes to get a design into silicon, through design, verification, and implementation. It stops short of software development and integration, which are steps that come with SoC Realization and System Realization. The output of Silicon Realization may be an analog, digital, or mixed-signal IP block that's later integrated into a SoC; an IC that does not require embedded software development; or an SoC ready for software integration.

As the electronics industry moves towards application-driven System Realization, and concerns rise about a growing profitability gap, the scope and complexity of Silicon Realization challenges will mushroom. Complexity will grow even more complex with the move to advanced process nodes such as 32nm and 22nm. Both the EDA industry and its customers must understand and respond to the challenges.

Since interfacing to the real-world requires analog circuitry, Silicon Realization nearly always involves a combination of analog and digital IP. But Silicon Realization goes far beyond the traditional view of “mixed-signal” design, which typically involves the importation of a few hard analog macros into a digital SoC. Silicon Realization involves the creation and integration of extremely large, complex analog and mixed-signal blocks—including blocks that were entire chips in previous process generations—into SoCs that support broad ranges of functionality. These days some analog/mixed-signal blocks may be programmable or configurable, as is the case with software-defined radio.

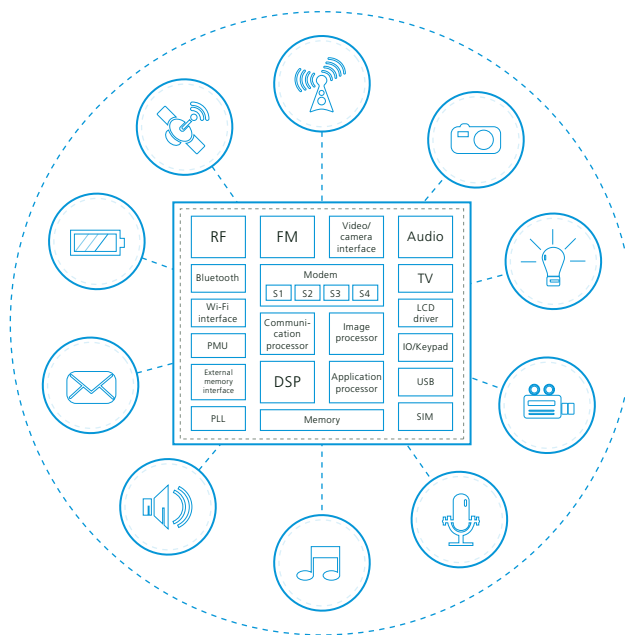


Figure 1 – Many SoCs today integrate large digital, analog, and mixed-signal blocks, including audio, video, and wireless interfaces that may have previously been entire chips.

Silicon Realization also involves the creation of full-custom digital, analog, and RF IP blocks and ICs. Concerns about productivity and profitability are just as acute for custom designers as they are in the digital world—and there’s far less design automation. Progress is needed in custom IC design as well, and some is already occurring. One example is the development of constraint-driven analog design flows.

While System Realization and SoC Realization are primarily tasks for integrators, both creators and integrators perform Silicon Realization. Thus, EDA360 must address the needs of creators by closing the productivity gap, and also address the needs of integrators by closing the profitability gap. The fundamental idea is to go from concept to silicon as quickly as possible and as cheaply as possible, while retaining the high quality that end consumers have come to know and demand.

Silicon Realization involves the creation and integration of extremely large, complex analog and mixed-signal blocks—including blocks that were entire chips in previous process generations—into SoCs that support broad ranges of functionality.

THREE CONCEPTS BEHIND SILICON REALIZATION

Three common themes are behind most of the approaches that are needed to close the productivity and profitability gaps in Silicon Realization:

Merge top-down design and bottom-up design. The traditional design approach is “bottom up”—designers start with the silicon and append the software and the applications later. The emerging application-driven approach starts with the application, defines the system, and then abstracts the hardware and software requirements from the system definition. But it’s not all top-down; information must flow upward from the silicon level as well. It is essential to understand power, performance, and cost at the system level, and this can happen only if information about silicon design, packaging, and manufacturing is made available at higher levels of abstraction.

Raise the level of abstraction. There are many ways to work at a higher, richer level of abstraction. The current move by digital design and verification teams to transaction-level modeling represents one approach, but there are others. For example, analog designers are using behavioral modeling languages such as Verilog-AMS, and can now simulate analog circuits in a digital environment using the “wreal” data type. However it’s done, higher levels of abstraction boost productivity, slash time-to-market, and lower costs.

Apply unified design intent. Whether for full-custom or digital design, a single, complete, and coherent representation of design intent will help avoid errors throughout the Silicon Realization flow. This unified design intent captures all aspects of the design—including function, constraints, and realization—to ensure everything is working towards a common goal and a common understanding. Working from this common design intent reduces the risks of specification misses and re-spins, and ensures that changes are tracked and implemented correctly and consistently throughout the design process.

CLOSING THE PRODUCTIVITY GAP IN SILICON REALIZATION

Closing the productivity gap requires innovative and differentiated capabilities in design, verification, and implementation. EDA360 needs to deliver the following capabilities to close the productivity gap.

DESIGN FOR PRODUCTIVITY

Raise the abstraction level early in the design process. For both analog and digital design, it is necessary to capture the design in as rich and high-level a form as possible, as early as possible. Then, tools can provide a greater degree of automation of subsequent design activity. This allows greater flexibility earlier in the design when critical design implementation decisions are made. With TLM, for example, digital designers are not locked into a micro-architecture, and automated SystemC high-level synthesis makes it easy to explore different choices and target key design requirements. On the analog side, using Verilog-AMS or wreal models helps designers capture design intent early and accelerate the verification cycle for IP integration.

Tackle multiple objectives simultaneously. Traditionally, engineers are forced to make “guesstimates” on how the final design will work, or focus on just a small number of design constraints. A better approach is to provide the early up-front data that’s needed to support informed decisions on both business and technical levels. Performance, power, yield, manufacturability, and cost must be considered concurrently rather than in isolation. Expanding on today’s chip planning tools, EDA360 tools will allow a rapid, concurrent “what-if” analysis enabled by higher levels of abstraction.

Whether for full-custom or digital design, a single, complete, and coherent representation of design intent will help avoid errors throughout the Silicon Realization flow.

Use a single source of design intent. There is often a loss of fidelity as one phase of silicon design is handed off to those working on the next phase. With a single, verified source of design intent, there is no loss of fidelity, nor is there a need for translation. In digital design, the “golden source” can reside in a number of different forms—such as TLM or register transfer level IP and Common Power Format models for capturing power intent. In an advanced mixed-signal design system, a common set of analog and digital constraints created at the design stage maintains intent throughout Silicon Realization.

VERIFY FOR PRODUCTIVITY

Holistic verification—use the best tool for the job. There are many different techniques for digital verification, including simulation, formal verification, and emulation. Approaches to analog simulation range from transistor-level SPICE simulation to analog behavioral modeling with the Verilog-AMS language. Working with the single goal of verifying the design intent, and utilizing a verification plan, EDA tools must choose the best approach for any given phase of the verification process and feed this back to the verification plan. The result is a holistic approach to verification using the most productive methods for each task.

Apply metric-driven, predictable verification. A modern IC design can contain more potential states than the number of atoms in the universe (roughly 10^{80}), so testing every possible combination is not feasible. What's needed is intelligent verification that "knows" what's been tested and what hasn't. A metric-driven, executable verification plan makes it possible to monitor and track testing through coverage metrics and determine what still needs to be verified. The same plan should also serve mixed-signal verification for analog and digital interfaces.

For both analog and digital design, it is necessary to capture the design in as rich and high-level a form as possible, as early as possible. Then, tools can provide a greater degree of automation of subsequent design activity.

Use the right verification abstraction. Abstraction is not only important for design—it is also critical for verification. When performing verification we need to use the most expressive description available, and provide just the right amount of detail to get the job done. For example, using a timed testbench and a netlist model would be a very inefficient way to perform traffic modeling in a network switch. Likewise, using a SPICE abstraction of a mixed-signal block during performance testing would lead to significant delays in the verification cycle.

IMPLEMENT FOR PRODUCTIVITY

Use single-pass design. Avoiding iterations requires a "right the first time" approach enabled by the feed-forward of important design data into each phase of the design implementation. As a given design element is implemented, enough information needs to be fed to the next design task to avoid potential problems and preserve critical assumptions from the previous steps. Even so, changes are inevitable. With an automated engineering change order (ECO) capability tied to formal verification, the design process can continue smoothly when a change occurs or a bug is fixed.

Understand realization. It is important to know how a final design will be implemented. If the design is an analog or digital IP block, designers should know what types of SoCs it might serve and what other IP it might be interacting with. If the final design will be packaged in a stacked-die (3D IC) configuration, this will influence many choices. In each step of the design process, data about the intended implementation must be made available at the appropriate levels of abstraction to make effective design tradeoffs.

Intelligently partition large digital blocks. At advanced silicon process nodes such as 32/28nm, a single digital IP block may contain millions of gates. We are truly entering the era of "Giga Gate, Giga Hertz" SoC design. This results in many challenges, including complex manufacturing rules, design data size explosion, design for test, yield ramp, variability, and many others. Large, digital design elements should not have to be artificially partitioned into smaller pieces due to tool restrictions or limitations in design expertise. EDA360 tools will expand capacity and scope, allowing designers to work in a more natural fashion, and to avoid many of the obstacles due to overly complex and mismatched physical and logical hierarchies in a design. This becomes especially important as higher levels of abstraction allow engineers to create significantly larger portions of a design.

CLOSING THE PROFITABILITY GAP IN SILICON REALIZATION

To close the profitability gap for integrators, EDA360 must support creation, integration, and optimization. This presents a somewhat different set of challenges and solutions compared to those for the productivity gap.

CREATE FOR PROFITABILITY

Use high levels of abstraction. The same prescription was given earlier for creators, but the focus for integrators is a little different. Here, the real concern is allowing integrators to focus on the value they really add by eliminating as many unnecessary steps as possible and reducing busywork. Creating or sourcing IP at the TLM level is one way to accomplish that goal.

Make IP reuse practical. Reuse is difficult and has not been cost-effective for most companies. It often costs more to reuse IP than to build it from scratch. Integrators who create IP must think carefully about where it will go, what applications it will serve, and what capabilities it does and does not need. They must provide appropriate documentation and deliverables, and pay attention to quality and configurability. Finally, the IP should be thoroughly verified so that re-verification is not needed during the integration phase.

Maintain design intent. A single understanding of design intent that is carried throughout the creation process will prevent costly errors. This understanding should be documented and executable by all parts of the design process. Leveraging greater automation based on this common design intent will have significant cost savings.

INTEGRATE FOR PROFITABILITY

Apply piece-by-piece signoff (“iterative correctness”). A significant portion of the overall design cost is due to redundant verification, or verification that is performed at an inefficient phase of the design. As each IP component is built or externally sourced, integrators should ensure it has the quality and capability to do the job. If components are properly validated and verified, integrators need only worry about the interfaces and higher-order functions when they assemble the components. There should be no need to re-verify the individual blocks, even those sourced externally.

A metric-driven, executable verification plan makes it possible to monitor and track testing through coverage metrics and determine what still needs to be verified. The same plan should also serve mixed-signal verification for analog and digital interfaces.

Use enablement services. A company that sources outside IP may temporarily need some specialized expertise they don’t normally have. For example, a digital design company might source some analog IP blocks and need help with mixed-signal integration and verification. There is significant cost and a recruitment challenge in building an internal team with enough understanding to satisfy this expertise requirement. A more cost-effective approach would be to take advantage of third-party services to fill this short-term expertise gap.

Offer visible integration. When analog blocks are brought into a digital SoC, or digital blocks are brought into a mostly analog IC, the imported analog or digital components are often “black boxes” with no visibility into internal structures. Or, they are fully elaborated models. This results in ineffective integration suffering from either too much visibility, which affects performance, or too little visibility, which impacts quality. Both tools and providers of IP should enable just enough visibility to ease the integration task.

OPTIMIZE FOR PROFITABILITY

Optimize for cost. While creators may emphasize technical differentiators such as power and performance, integrators are deeply concerned about the cost of design. They need to select the most cost-efficient implementation and avoid any possible over-design. The traditional approach involves overdesigning with the significant use of “guardbands” to minimize potential problems and closure issues. For example, guardbands may require a timing margin as high as 30 to 50 percent, forcing the chip design to support a much broader-than-needed range of timing performance or variability, which in turn introduces poor quality, significant waste, and lost profitability per part. By leveraging the characteristics such as what-if analysis based on higher levels of abstraction, single pass design, and implementation awareness, the need for this kind of over-design can be avoided.

Reuse is difficult and has not been cost-effective for most companies. It often costs more to reuse IP than to build it from scratch.

Manage changes. Changes are inevitable, and it is the management of this change that can make the difference between a profitable design released on schedule, and a design that fails to make it to market. As such, it is important to simplify and manage change within the design process. For integrators, this is true not only for changes made by their own team, but by any external teams or suppliers from which IP was sourced. When changes happen, their impact should be minimized, localized, and tracked through to completion using a range of capabilities such as enterprise management, ECO automation, and metric-driven verification. Managing change effectively is managing profitability.

Consider software as a service. Many design teams struggle to deploy a design environment even when using a reference flow defined by their production partner. Much of this difficulty is centered around the lack of appropriate expertise and subtle differences in infrastructure. Using Software-as-a-Service allows a design environment to be fully configured and externally hosted using either dedicated third-party servers or “cloud” computing. SaaS can help integrators leverage the capabilities they need when they need it. This alone can greatly reduce design cost and accelerate profitability. By making this environment production-aware, the environment is optimized not just for the customer, but for the foundry that is manufacturing the design. The result is a highly optimized design process all the way from design to manufacturing.

In brief, the best way to address profitability is to have a top-down view of what you’re trying to build, automate as many details as possible, and reuse verified components that can simply be snapped together. Lego blocks provide a good analogy. With Lego blocks, you start by envisioning a project such as a city street. You join multiple small objects together to build bigger objects, like houses. Then you can place multiple houses on a street. But the starting point is still the large-scale vision of what you plan to build.

CONCLUSION

Silicon Realization has been around for decades, but when we look at it through the eyes of both creators and the new breed of integrators, a new perspective emerges. While traditional EDA has long focused on aspects of Silicon Realization, the ever-increasing scope of challenges means there is more work than ever to close the productivity gap. For example, EDA360 seeks to raise the abstraction level for both analog and digital design; use single sources of design intent; employ metric-driven verification; use single-pass, correct-by-construction design; and provide intelligent partitioning at the logical and physical levels.

The newer challenge is helping integrators close the profitability gap. Here, EDA360 will provide solutions that make IP reuse easier, allow “iterative correctness,” optimize cost, and manage change across external suppliers and internal teams. But EDA360 is not just about tools and technology; it also expands into business and engagement models. Software-as-a-Service models, for example, will help integrators improve the design process and control costs, while targeted services engagements will provide access to expertise on an as-needed basis.

With this chapter, we end this Vision Paper where EDA typically begins—the production of cost-effective, power-efficient silicon. But for us, it is the end of a journey that began with end-user demand for multiple, concurrent software applications; anywhere, anytime connectivity; and audio, video, and 3D graphics. We have seen how the development of ever-more creative applications is driving system development and giving rise to a new class of innovators called “integrators.” We have shown how the needs of integrators are different from those of the creators the EDA industry has always served.

As with any crisis or inflection point, there is both danger and opportunity for existing providers. At Cadence, we see the opportunity. We plan to be leaders in a revitalized EDA industry that will serve both creators and integrators. We will work with our ecosystem partners to provide the necessary support for System, SoC, and Silicon Realization, and to enable a new era of application-driven design. Please join us and let us know what you think at www.cadence.com/eda360.



Corporate Headquarters

2655 Seely Avenue San Jose, CA 95134

Phone: 408.943.1234

Fax: 408.428.5001

www.cadence.com/eda360