Exploiting Core’s Law: Get “More than Moore” productivity from your ASIC and SOC Design Teams

In 1965, Gordon Moore prophesized that integrated circuit density would double roughly every one to two years. The universal acceptance and relentless tracking of “Moore’s Law” set a grueling pace for chip developers. Moore’s Law makes transistors ever cheaper and faster (good) while inviting system buyers to expect constant improvements to end-product functionality, battery life, throughput, and cost. The moment a new function is technically feasible, the race is on to deliver it. Today, it is perfectly feasible to build ASIC devices with more than 100 million transistors, and it won’t be long before we see billion-transistor chips that combine processors, memory, logic, and interface components. Designing such chips is an increasingly big challenge. MP ASICs or MPSOCs, multiprocessor SOCs, help you achieve “more than Moore” design productivity gains by leveraging proven IP blocks that give your design team a big head start in the race to design that next killer ASIC.

High ASIC integration creates terrific opportunities. The remarkable characteristics of CMOS silicon scaling allow the cost, size, performance, and power for a given function to all improve simultaneously. This scaling allows continuous end-product improvements: longer battery life, smaller size, more functionality, and better user productivity. ASIC semiconductor scaling has been a key driver for the parallel revolutions in digital consumer electronics, personal computing, and the Internet. Most observers expect this scaling trend to continue for at least another 10 years.

The growth in available on-chip transistors creates a fundamental role for concurrency in ASIC designs. Concurrent tasks such as audio and video processing and network-protocol stack management can operate largely independently of one another. Complex tasks with inherent internal execution parallelism can be decomposed into a tightly-coupled collection of sub-tasks operating in parallel to perform the same work as the original non-parallel task implementation. Such tasks are compositionally concurrent and this kind of concurrency promises significant improvements in application latency, data bandwidth, and energy efficiency when compared to serial execution of the same collection of tasks using a single computational resource. In other words, large numbers of transistors lead to the inclusion of multiple execution engines, which in turn allow design teams to exploit many forms of concurrency that were previously out of reach.

Massive silicon integration presents a terrific opportunity but the associated design task is correspondingly terrifying. Three forces work together to make ASIC design tougher and tougher. Semiconductor manufacturers have had astonishing success in tracking Moore’s Law. They truly have given ASIC designers twice as many gates to play with every two years. Second, the continuous improvement in process geometry
and circuit characteristics motivates chip builders to design with new IC fabrication
technologies as they come available to cut costs and add features. Third, and perhaps
most important, the end markets for electronic products—consumer, computing, and
communications systems—are in constant churn. System vendors require a constant
stream of new functions and performance to justify new consumer purchases.

As a result, the design “hill” keeps getting steeper. Certainly, improved chip-design
tools help—faster RTL simulation and verification, higher capacity logic synthesis,
and better block placement and routing all mitigate some of the difficulties.
Similarly, increasing logic design reuse (IP reuse) reduces the amount of new design
that must be done for each chip.

But all these improvements fail to close the design gap. This well-recognized
phenomenon is captured in the International Technology Roadmap for
Semiconductors. A simple comparison of the growth in logic complexity versus
designer productivity appears in Figure 1 below. The figure shows that the industry’s
adherence to Moore’s Law put ASIC transistor count on an aggressive growth rate of
58% CAGR over the last few decades. Meanwhile, designer productivity improved,
but far more slowly with only a 20% CAGR.

![Figure 1: Design Complexity versus Staff Productivity – Long-Term Trends](image)

Even as designers wrestle with the growing resource demands of advanced chip
design, they face two key questions in the quest to build profitable ASICs:

1. How do design teams ensure that the chip specification really satisfies
customer needs?
2. How do design teams ensure that the chip really meets those specifications?
If it has sufficient time to think about it, a good design team anticipates future needs of current customers and potential future customers—it has a built-in road map. However, there are roadblocks on these road maps.

**Roadblock 1: Building the Wrong Chip (Inflexibility)**

If the ASIC design team fails to adequately answer the first question listed above, the resulting chip may work perfectly but sales will not be sufficient to justify the design expense and manufacturing effort. Specific key-customer demands or market trends—such as the emergence of new data-format standards or new feature expectations across an entire product category—can cause requirements to change in mid project. Firmware is a very effective way of solving problems or adding features to a system however most ASIC designs only include some form of embedded control processor with limited performance, which often precludes these control processors from being used for essential data-processing tasks. In such situations, firmware usually cannot be used to change features or add fundamental new ones.

**Roadblock 2: Building the Chip Wrong (Failed Design Process)**

If the ASIC design team fails to adequately answer the second question listed above, the team must expend additional time and resource to fix the design. This resource diversion delays market entry and often cause companies to miss key customer commitments. The failure is most often realized as a program delay. This delay may come in the form of missed integration or verification milestones, or it may come in the form of hardware bugs—explicit logic errors that are not caught in the limited verification coverage of typical hardware simulation. Here are the dismal statistics for ASIC designs:

- 24% of all projects are canceled due to schedule slip
- 54% of all projects are completed late
- 33% of all projects miss functional or performance targets
- 80% of the effort needed to correct errors is for errors discovered late in the project

The underlying cause for these poor project results might be as trivial as a subtle error in a single design element. A miscommunication of requirements—subtle differences in assumptions between hardware and software teams, between design and verification teams, or between SOC designer and SOC library or foundry supplier—could also be the root cause. In any case, project slips usually end up forcing the ASIC design team into a frantic cycle of re-design, re-verification, and chip re-fabrication. These design “spins” often take six months or more and significantly disrupt product and business plans.

---

To improve the design process, you must consider simultaneous changes in all three interacting dimensions of the design environment: the design elements, the design tools, and the design methodology.

- Design elements are the basic ASIC building blocks: the silicon structures and the logical elements that form the basic vocabulary of design expression. Historically, these blocks have been basic logic functions (NAND and NOR gates and flip-flops) plus algorithms written in C and assembly code running on embedded processor cores or digital signal processors (DSPs).

- Design tools are the application programs and techniques that ASIC designers use to capture, verify, refine, and translate design descriptions for particular tasks and subsystems. Historically, tools such as RTL compilation and verification, code assemblers and compilers, and standard-cell placement and routing tools have comprised the essential tool box for complex chip design.

- Design methodology is the design team’s strategy for combining the available design elements and tools into a systematic process to implement the target silicon and software. A design methodology specifies what elements and tools are available, describes how to use the tools at each step of the design refinement, and outlines the sequence of design steps. The current, dominant ASIC design methodology is built around four major steps, typically implemented in the following order:

1. hardware/software partitioning
2. detailed RTL block design and verification
3. chip integration of RTL blocks
4. processors, memories, and hardware/software integration

Changes in any one dimension are unlikely to prevent the pitfalls of ASIC design—namely “building the wrong chip” or “building the chip wrong.” Piecemeal improvements made to RTL design or software development tools will not solve the larger system-design problem. Instead, your design team must reframe the design problem itself. The design elements, the key tools, and the surrounding methodology must all change together.

The Fundamental Trends of ASIC and SOC Design

Several basic trends strongly indicate that the engineering community needs a new approach to ASIC and SOC design. The first such trend is the seemingly inexorable growth in silicon density, which underlies the fundamental economics of building electronic products in the 21st century. At the center of this trend is the fact that the semiconductor industry seems willing and able to continue to push chip density by consistent, sustained innovation through smaller transistor sizes, smaller interconnect geometries, higher transistor speed, significantly lower cost, and lower power dissipation over a long period of time.
Technical challenges for semiconductor scaling abound, as they have for the last three decades. Issues of power dissipation, nanometer lithography, signal integrity, and interconnect delay all will require significant process innovation. However, 50 years of past experience suggests that these challenges, at worst, will only marginally slow down the pace of scaling.

The central question remains this: *How will we design what Moore’s Law scaling makes technically achievable?*

Silicon scaling stimulates the second trend—the drive to take the available transistor density and integrate the enormous diversity and huge number of functions required by modern electronic products into one piece of silicon. The increasing integration level creates the possibility of taking all the key functions associated with a network switch, or a digital camera, or a personal information appliance, and putting these functions—all of the logic, all of the memory, all of the interfaces, in fact almost everything electronic in the end-product—into one piece of silicon, or something close to it.

If your team won’t or can’t do this, another team will find a way.

The benefits of high silicon integration levels are clear. Tight integration drives the end product’s form factor and produces complex systems small enough to put into your pocket, inside your television, or in your car. High integration levels also reduce power dissipation, allowing more end products to be battery powered, fan-less, or usable in a much wider variety of environments. Ever increasing integration levels drive raw performance—in terms of how quickly a product will accomplish tasks or in terms of the number of different functions that a product can incorporate—ever upward, ideally enough to make the average consumer rush to their favorite retailer to buy the new products even if it means replacing old ones to get the new features.

**A New ASIC for Every System is a Bad Idea**

The silicon specialization that results from higher and higher integration creates an economic challenge for the ASIC development team. If all of the electronics in a system are embodied in roughly one chip, that chip is increasingly likely to be a direct reflection of the end product the designer is trying to define. Such a chip design could lack flexibility. If so, it cannot be used in a wide variety of products.

In the absence of some characteristic that makes that highly integrated chip significantly more flexible and reusable, ASIC design trends towards a direct 1:1 correspondence between chip design and system design. If ASIC design continues to go down this road, the time needed for new system development and the amount of engineering resources required to develop a new system will, unfortunately, become
at least as great as the time and costs to build new chips. That’s an unnecessary restriction.

In the past, product designers built systems by combining chips on large printed circuit boards (PCBs). Different systems used different combinations of (mostly) off-the-shelf chips soldered onto system-specific PCBs. This design approach worked, and worked very well for the time, because a wide variety of silicon components were available and because PCB design and prototyping was easy. System reprogrammability was less important because system redesign was relatively cheap and quick.

In today’s world of nanometer system-on-silicon technology, the situation is dramatically different. Demands for smaller physical system size, greater energy efficiency, and lower manufacturing cost have all made the large system PCB obsolete. Volume-oriented end-product requirements can only be satisfied with system-on-chip designs. Even when appropriate “virtual components” are available as ASIC building blocks, system design integration and prototyping on a chip are more than two orders of magnitude more expensive than PCB design and prototyping. Moreover, ASIC design changes take months while PCB changes take days (sometime just one day).

So here’s the paradox: SOC design is mandatory to reap the benefits of nanometer silicon but to make SOC design practical, SOCs cannot be built like PCBs. The design elements must address the problem of SOC inflexibility. Chip-level inflexibility is really a reusability crisis with respect to the chip’s hardware design. Despite substantial industry attention to the benefits of block-level hardware reuse (IP reuse), the growth in internal complexity of blocks coupled with the complex interactions among blocks has limited the systematic and economical reuse of IP blocks.

Too often end-customer requirements, implemented standards, and the necessary interfaces to other functions must evolve with each product variation. These boundaries constrain successful block reuse to two categories:

1. simple blocks that implement stable interface functions
2. inherently flexible functions that can be implemented in processors, whose great flexibility and adaptability are realized via software programmability

On the other hand, a requirement to build new chips for every system would be an economic disaster for system developers because there’s no question that building chips is hard. We can improve the situation somewhat with better chip-design tools, but in the absence of significant innovation in chip-design methodology, the situation’s not getting much better, at least it’s not getting better very fast.

In fact, it would appear that in the absence of some major innovation, the efforts required to design a chip will increase more rapidly than the transistor complexity of the chip itself. We’re literally losing ground in systems design because innovation in design methodology is lacking. However, we cannot afford to lose ground on this problem. As system and chip design grow closer together, we need to gain ground.


**ASIC Design Reform: Lower Design Cost and Greater Design Flexibility**

System developers are trying to solve two closely related problems:

- To develop system designs with significantly fewer resources by making it much, much easier to design the chips in those systems.
- To make ASICs more adaptable so not every new system design requires a new chip design.

The way to solve these two problems is to make the ASIC sufficiently programmable so that one chip design will efficiently serve 10, or 100, or even 1000 different system designs while giving up none or perhaps just a few of the benefits of integration. Solving these problems means having chips available off the shelf to satisfy the requirements of the next system design and amortize the costs of chip development over a large number of system designs. This concept is central to the development of the application-specific standard part (ASSP).

The market and technology trends discussed above constitute the forces behind the need for a fundamental shift in IC design. That fundamental shift will ideally provide both a big improvement in the effort needed to design SOCs (not just in the silicon but also the required software) and it will increase the intrinsic flexibility of SOC designs so that the design effort can be shared across many system designs.

Economic success in the electronics industry hinges on the ability to make future ASICs more flexible and more highly optimized at the same time. The core dilemma for the semiconductor industry and for all the users of ASIC devices is really simultaneous management of flexibility and optimality.

ASIC developers are trying to minimize chip design costs and trying to get closer and closer to the promised benefits of high-level silicon integration at the same time. Consequently, they need to take full advantage of high-density nanometer silicon and, at the same time, they need to overcome or mitigate the issues created by the sheer complexity of those nanometer ASIC designs and the high costs and risks associated with long ASIC development cycles. Programmability allows ASIC designers to substantially mitigate the costs and risks of complex SOC designs by accelerating the initial development effort and by easing the effort to accommodate subsequent revisions of system requirements.
Programmability

Rising system complexity also makes programmability essential to successful ASIC designs. The more efficient programming becomes, the more pervasive it will be. The market has already endorsed the concept of programmability in a number of ways including field-programmable gate arrays (FPGAs), standard microprocessors and processor cores coupled with firmware reprogrammability through Flash memory, and reconfigurable logic.

Programmability offers benefits at two levels. First, programmability increases the likelihood that a pre-existing design can meet the performance, efficiency, and functional requirements of the system. If there is a fit, no new ASIC development is required—an existing platform will serve with some firmware reprogramming. Second, programmability means that even when a new ASIC must be designed, more of the total functions are implemented in a programmable fashion, reducing the design risk and effort. The success of both the FPGA and processor markets can be traced to these factors.

The programming models for different system platforms differ widely. Traditional processors (including DSPs) can execute applications of unbounded complexity. However, as complexity grows, performance typically suffers. RISC processors use sophisticated pipelining and circuit-design techniques to achieve high clock frequency, but in the end they achieve only modest parallelism—one (or a few) operations per clock cycle. All FPGAs, by contrast, have finite capacity—one the problem grows beyond some complexity level, the problem will not fit in that FPGA at all. On the other hand, FPGAs can implement algorithms with very high levels of intrinsic parallelism, sometimes performing the equivalent of many hundreds of operations per cycle. Technical limitations force FPGAs to operate at more modest clock rates than processors and FPGAs tend to have significantly larger die sizes and higher chip costs than processors used in the same applications.

Programmability versus Efficiency

The various flavors of programmability allow the underlying silicon design to be somewhat generic while permitting configuration or personalization for a specific situation at the time that the system is booted or during system operation. The traditional problem with programmability is that there’s a tremendous gap in efficiency and/or performance between a hard-wired design and a design with the same function implemented with programmable technology. This gap can be called “programmability overhead.”

This overhead may be seen as the increased area for implementation of a function using programmable methods, compared to a hardwired implementation with the same performance. Alternatively, the overhead can take the form of an increase in execution time for a programmable design solution when compared to a hardwired implementation that consumes the same silicon area. As a rule of thumb, the overhead for FPGA or generic processor programmability is more than a factor of ten, and can be as much as a factor of one hundred. For example, hardwired logic solutions are typically about 100x faster for security applications such as DES and
AES encryption than the same tasks implemented with a general-purpose RISC processor. An FPGA implementation of these encryption functions may run only at 3-4x lower clock frequency than hardwired logic, but may require 10-20x more silicon area.

These inefficiencies stem from the excessive generality of the universal, reprogrammable digital substrates: FPGAs and general-purpose processors. The designers of these general-purpose substrates attempt to construct platforms that cover all possible scenarios. Unfortunately, creating truly general-purpose substrates requires a superabundance of basic facilities—from which to fabricate specific computational functions—and connection paths to move data among computation functions. Silicon efficiency is constrained by the limited reuse or “time-multiplexing” of the transistors implementing an application’s essential functions.

In fact, if you look at either an FPGA or a general-purpose processor performing an add computation, you will find a group of logic gates comprising an adder surrounded by a vast number of multiplexers and wires to deliver the right data to the right adder at the right moment. The circuit overhead associated with storing and moving the data and selecting the correct sequence of functions leads to much higher circuit delays and a much larger number of required transistors and wires than a design where the sequence of operations to be performed is known.

General-purpose processors rely on time-multiplexing of a small and basic set of function units for basic arithmetic and logical operations and memory references. Most of the processor logic serves as hardware to route different operands to the small set of shared hardwire function units. Communication among functions is implicit in the reuse of processor registers and memory locations by different operations. FPGA logic, by contrast, minimizes the implicit sharing of hardware among different functions. Instead, each function is statically mapped to particular region of the FPGA silicon, so each transistor typically performs a single function repeatedly. Communication among functions is explicit in the static configuration of interconnect among functions.

The more that is known about the required computation, the more the transistors involved in the computation can be interconnected with dedicated wires to improve hardware utilization. Both general-purpose processors and general-purpose FPGA technologies have overhead, but an exploration of software programmability highlights the hidden overhead of field hardware programmability.

Modern software programmability’s power really stems from two complementary characteristics. One of these is abstraction. Software programs allow developers to deal with computation in a form that is more concise, more readily understood at a glance, and more easily enhanced independent of implementation details. Modest-sized software teams routinely develop, reuse, and enhance applications with hundreds of thousands of lines of source code, including extensive reuse of operating systems, application libraries, and middleware software components. In addition, sophisticated application-analysis tools have evolved to help teams debug and maintain these complex applications.
By comparison, similar-sized hardware teams consider logic functions with tens of thousands of lines of Verilog or VHDL code to be quite large and complex. Such hardware blocks are modified only with the greatest care.

The second characteristic is software’s ease of modification. System functionality changes when you first boot the system and it changes dynamically when you switch tasks. If a task requires a complete change to a subsystem’s functionality, the system can load a new firmware-based personality for that subsystem from memory in a few microseconds in response to changing system demands.

This is a key point: the economic benefits of software flexibility appear both in the development cycle (what happens during the time between product conception and first system “power-on”) and during the expected operational life of a design (what happens during the time between freezing the product specification and the moment when the last variant of the product is shipped to the last customer).

**The Path to Profit**

Continuous adaptability to new product requirements plays a central role in improving product profitability. If the system can be reprogrammed quickly and cheaply, developers reduce the risk of failing to meet design specifications and have greater opportunity to quickly adapt the product to new customer needs. Field-upgrading software has become routine with PC systems and the ability to upgrade firmware in the field is starting to find its way into embedded products. For example, products such as Cisco network switches get regular software upgrades. Greater flexibility (at a given cost) increases the number of customers, increases ASIC sales volume, and lowers ASIC unit costs.

In contrast, hardwired design choices must be made very early in the system-design cycle. If, at any point in the development cycle—during design, during prototyping, during field trials, during upgrades in the field, or during second-generation product development—the system developers decide to change key computation or communication decisions, it’s back to square one for the system design. Hard-wiring key design decisions also narrows the potential range of customers and systems into which the ASIC might fit and limits the potential volume shipments.

Making systems more programmable has both benefits and liabilities. Benefits include agility and efficiency. Designers don’t need to decide exactly how the computational elements relate to each other until later in the design cycle, when the cost of change is low. Whether programmability results from a net list being loaded into an FPGA or firmware into processors, the designer need not decide on a final configuration until system power up. If programmability is realized through firmware running on processors, developers can defer many design decisions until system bring-up. Some decisions can be deferred until the eve of product shipment.

The liabilities of programmability have historically been cost, power efficiency, and performance. Conventional processor and FPGA programmability carries an overhead of thousands of transistors and thousands of microns of wire between the
computational functions. This overhead translates into large die size and low clock frequency for FPGAs and long execution time for processors, when compared to hardwired logic implementing the same function. Freezing the hardware implementation (removing programmability) can dramatically improve unit cost and system performance but dramatically raises design risk and design cost. Consequently, the design team faces trying choices. Which functions should be implemented in hardware? Which in software? Which functions are most likely to change? How will communication among blocks evolve? Figure 2 gives a conceptual view of this tradeoff.

Figure 2: Improving the ASIC Design Style Adds Programmability and Flexibility

The vertical axis in Figure 2 indicates the intrinsic complexity of a block or of the whole system. The horizontal axis indicates the performance or efficiency requirement of a block or of the entire system. One recurring dilemma for ASIC design is: flexible design solutions that support complex designs sacrifice performance, efficiency, and throughput; solutions with high efficiency or throughput often sacrifice flexibility.

The two curves shown in Figure 2 represent two overall design styles or methodologies. Within a design methodology, a variety of solutions are possible for each block or for the whole system but the design team must always trade off complexity and high efficiency. An improved design style or methodology still must deal with tradeoffs, but there’s an overall improvement in the “flexibility-efficiency product.” In moving to an improved design methodology, the ASIC design team can focus on improving programmability—to get better flexibility at a given level of performance—or it can focus on improving performance—to get better efficiency and throughput at a given level of programmability.
In a sense, the key to efficient ASIC design is managing uncertainty. If the design team can optimize all dimensions of the product for which requirements are stable and leave flexible all dimensions of the product that are unstable, they will have a cheaper, more durable, and more efficient product than their competitors. Their company will sell more ASICs.

**The Key to ASIC Design Success: Domain-Specific Flexibility**

The central goal of ASIC development is to strike an optimal balance between getting just enough flexibility to meet changing demands still realizing the efficiency and optimality associated with targeting an end application. Therefore, what’s needed is an ASIC design methodology that permits a high degree of system and subsystem parallelism, an appropriate degree of programmability, and rapid design.

It is not necessary for ASIC-based system developers to use a completely universal piece of silicon—most ASICs ship in enough volume to justify specialization. For example, digital cameras don’t use the same chip that’s used in a high-end optical network switch. One camera chip, however, can support a range of related consumer imaging products, as shown in Figure 3.

**Figure 3: One Flexible Imaging ASIC Can Serve Many End Products**

The difference in benefit derived from a chip shared by ten similar designs, versus one shared by 1,000 designs, is relatively modest. If each camera design’s volume is 200,000 units, and the shared ASIC design costs $10M, then the ASIC design costs contribute $5 to final camera cost (~5%). Sharing the ASIC design across 1000
designs could save $5 in amortized design cost, but the resulting ASIC design would almost certainly require such generality that chip production costs would increase by far more than $5. ASIC designs need not be completely universal—high-volume end products can easily afford to use chip-level design platforms that are appropriate to their application domain, yet flexible within it.

If designers have sufficient flexibility within an ASIC to adapt to any tasks they are likely to encounter during that design’s lifetime, then they essentially have all the relevant benefits of universal flexibility without incurring much of the overhead of universal generality. If the platform is designed correctly, the cost for application-specific flexibility is much lower than universal flexibility realized in a truly universal device such as an FPGA or a high-performance, general-purpose processor.

In addition, a good design methodology should enable as broad a population of hardware and software engineers as possible to design and program the ASICs. The larger the talent pool, the faster the development and the lower the project cost.

The key characteristics for such an SOC design methodology are:

1. support for concurrent processing,
2. appropriate application efficiency, and
3. ease of development by people who are not necessarily ASIC design specialists

An Improved ASIC Design Methodology

A fundamentally new way to speed development of nanometer, mega-gate ASICs is emerging. First, processors replace hardwired logic to accelerate hardware design and bring full chip-level programmability. Second, those processors are extended, often automatically, to run functions very efficiently with high throughput, low power dissipation, and modest silicon area. Blocks based on extended processors often have characteristics that rival those of the rigid, non-programmable RTL blocks they replace. Third, these processors become the basic building blocks for complete ASICs. As a result, rapid development, flexible interfacing, and easy programming and debugging of the processors accelerate the overall design process.

Finally, and perhaps most importantly, the resulting ASIC-based end products are both efficient and highly adaptable to changing requirements. This improved ASIC design flow allows full exploitation of the intrinsic technological potential of nanometer silicon—parallelism, pipelining, fast transistors, and application-specific operations—and reaps the benefits of modern software development. A sketch of this new ASIC design flow appears in Figure 4.

Note: A key benefit of this design flow is that team members need not learn many new design techniques. The concepts of IP block reuse and firmware-driven functional definition are already well understood.
The new design flow starts with the high-level system requirements, especially the external input and output requirements for the new ASIC platform and the set of tasks that the system performs on the data flowing through the system. The computation within tasks and the communication among tasks and interfaces are optimized using application-specific processors and fast tools for analyzing function, performance, and cost.

A key benefit to this new design flow is that the design team gets an accurate system model early in the design schedule so detailed hardware and software design can proceed in parallel. Early and accurate modeling of both hardware and software reduces development time and minimizes the likelihood that expensive surprises will materialize late in the project.

Using this design approach means that designers can move through the design process with fewer dead ends and false starts and without the need to back up and start over. It means that ASIC designers can make a much fuller and more detailed exploration of the design possibilities early in the design cycle. As a consequence, the design team will better understand the design’s hardware costs, application performance, interface, and programming model.
Taking this approach to designing SOCs means that the explored design space will be as large as possible with the fewest compromises in the cost and the power efficiency of that platform. The more a design team uses the application-specific processor as the basic ASIC building block—as opposed to hard-wired logic written as RTL—the more the ASIC will be able to exploit the flexibility inherent in a software-centric design approach.

The Configurable Processor as Building Block

The basic building block of this methodology is a new type of microprocessor: the configurable, extensible microprocessor core. These processors are created by a generator that automatically transforms high-level, application-domain requirements (in the form of instruction-set descriptions or even examples of the application code) into an efficient hardware design and associated software-development tools. The MP (multiprocessor) approach to ASIC and SOC design allows engineers without microprocessor design experience to specify, evaluate, configure, program, interconnect, and compose those basic building blocks into combinations of processors that together create the ASIC’s digital core.

To develop a processor configuration using one of these configurable microprocessor cores, the chip designer or application expert uses the processor-generator interface (shown in Figure 5) to select or describe the application source, instruction-set options, memory hierarchy, closely-coupled peripherals, and interfaces required by the application.

Figure 5: Basic Processor Generation Flow
It takes about one hour to fully generate the hardware design—which takes the form of a hardware description in a standard RTL language, EDA tool scripts and test benches, and the associated software-development environment (C and C++ compilers, debuggers, simulators, RTOS code, and other support software). This timely delivery of the hardware and software infrastructure permits rapid tuning and testing of the target applications on that processor design.

Application-specific processors perform all of the same tasks that a microcontroller or a high-end RISC processor can perform: they run applications developed in high-level languages; implement a wide variety of real-time features; and support complex protocols stacks, libraries, and application layers. Application-specific processors perform generic integer tasks very efficiently, even as measured by traditional microprocessor power, speed, area, and code-size criteria. But because application-specific processors can incorporate the wide data paths, instructions, and register storage for idiosyncratic data types and computation required by embedded applications, they can support virtually all of the functions that chip designers have historically implemented as hard-wired logic—state machines and data paths.

The Transition to MPSOC Design

The transition from conventional ASIC design methodology to a new multiple-processor SOC (MPSOC) design methodology offers two fundamental benefits. First, it brings flexibility and speed of design to traditionally hardwired functions. The performance and energy efficiency of configurable processors far exceed that of conventional processors and rivals capability of hardwired logic functions, but with simpler, faster initial design and post-silicon programmability through firmware. Automatically generated processors have already displaced hardwired RTL design in complex, data-intensive function blocks for hundreds of ASIC designs.

Second, the pervasive use of configurable processors as the basic building-block of choice simplifies system-level design. (This MPSOC characteristic is increasingly called “Core’s Law” to correspond to Moore’s Law.) Across all the functions implemented with application-specific processors, hardware and software teams use one set of hardware interfaces, software tools, simulation models, and debug methods. This unification reduces design time, misunderstandings between hardware and software developers, and risk of failure.

Your next ASIC design project need not be affected by the dismal completion statistics quoted at the beginning of this white paper. Your team risks those statistics if it insists on using old design techniques to create nanometer ASICs. However, there are ways to update your team’s playbook. If you would like help optimizing your next SOC design, contact Tensilica for a consultation.
US Sales Offices:
Santa Clara, CA office:
3255-6 Scott Blvd.
Santa Clara, CA 95054
Tel: 408-986-8000
Fax: 408-986-8919

San Diego, CA office:
1902 Wright Place, Suite 200
Carlsbad, CA 92008
Tel: 760-918-5654
Fax: 760-918-5505

Boston, MA office:
25 Mall Road, Suite 300
Burlington, MA 01803
Tel: 781-238-6702 x8352
Fax: 781-820-7128

International Sales Offices:
Yokohama office (Japan):
Xte Shin-Yokohama Building 2F
3-12-4, Shin-Yokohama, Kohoku-ku,
Yokohama
222-0033, Japan
Tel: 045-477-3373 (+81-45-477-3373)
Fax: 045-477-3375 (+81-45-477-3375)

UK office (Europe HQ):
Asmec Centre
Eagle House
The Ring
Bracknell
Berkshire
RG12 1HB
Tel: +44 1344 38 20 41
Fax: +44 1344 30 31 92

Israel:
Amos Technologies
Moshe Stein
moshe@amost.co.il

Beijing office (China HQ):
Room 1109, B Building, Bo Tai Guo Ji,
122th Building of Nan Hu Dong Yuan, Wang Jing,
Chao Yang District, Beijing, PRC
Postcode: 100102
Tel: (86)-10-84714323
Fax: (86)-10-84724103

Taiwan office:
7F-6, No. 16, JiHe Road, ShihLin Dist,
Taipei 111, Taiwan ROC
Tel: 886-2-2772-2269
Fax: 886-2-66104328

Seoul, Korea office:
27th FL., Korea World Trade Center,
159-1, Samsung-dong, Kangnam-gu,
Seoul 135-729, Korea
Tel: 82-2-6007-2745
Fax: 82-2-6007-2746

Beijing office (China HQ):
Room 1109, B Building, Bo Tai Guo Ji,
122th Building of Nan Hu Dong Yuan, Wang Jing,
Chao Yang District, Beijing, PRC
Postcode: 100102
Tel: (86)-10-84714323
Fax: (86)-10-84724103

Taiwan office:
7F-6, No. 16, JiHe Road, ShihLin Dist,
Taipei 111, Taiwan ROC
Tel: 886-2-2772-2269
Fax: 886-2-66104328

Seoul, Korea office:
27th FL., Korea World Trade Center,
159-1, Samsung-dong, Kangnam-gu,
Seoul 135-729, Korea
Tel: 82-2-6007-2745
Fax: 82-2-6007-2746