CHAPTER 1

INTRODUCTION TO 21ST-CENTURY
SOC DESIGN

The past is prologue for the future
—common saying, frequently ignored

Systems-on-chips (SOCs) are, by definition, electronic systems built on single chips. Also by definition, every SOC incorporates at least one microprocessor. Some SOCs use two or three microprocessors to accomplish required tasks and a few SOC designs employ many dozens of processors. To see how SOC design got where it is today and where market and technological forces will take it tomorrow, we start by first looking at how electronic system design has evolved since the microprocessor’s introduction.

1.1 THE START OF SOMETHING BIG

The course of electronic systems design changed irreversibly on November 15, 1971, when Intel introduced the first commercial microprocessor, the 4004. Before that date, system design consisted of linking many hardwired blocks, some analog and some digital, with point-to-point connections. After the 4004’s public release, electronic system design began to change in two important ways.

First, and most obvious, was the injection of software or firmware into the system-design lexicon. Prior to the advent of the microprocessor, the vast majority of system designers had only analog and digital design skills. If they had learned any computer programming, it was used for developing design-automation aids or simulation programs, not for developing system components. After the Intel 4004’s introduction, system developers started to learn software programming skills, first in assembly language and then in high-level languages such as PL/1, Pascal, and C as compilers got better and memory became less expensive.

The other major change to system design caused by the advent of the microprocessor—a change that’s often overlooked—is the use of buses to interconnect major system blocks. Figure 1.1 shows a block diagram of a Hewlett-Packard 3440A digital voltmeter that was designed in 1963, eight
FIGURE 1.1

A digital voltmeter block diagram adapted from the design of a HP 3440A, circa 1963.
years before the microprocessor appeared. This block diagram, typical of
the era, shows a mixture of analog and digital elements interconnected
with point-to-point connections. Even the all-digital measurement counter,
which stores the result of the analog-to-digital conversion, consists of four
independent decade counters. Each decade counter communicates to the
next counter over one wire and each counter drives its own numeric dis-
play. There are no buses in this design because none are needed. Note that
there are no microprocessors in this design either. Microprocessors wouldn’t
appear for another eight years.

Figure 1.2 illustrates how a system designer might implement a digital
voltmeter like the HP 3440A today. A microprocessor controls all of the
major system components in this modern design implementation. One
key change: the processor communicates with the other components over
a common bus—the microprocessor’s main bus.

From a systems-design perspective, there are significant differences
between the digital voltmeter design from the early 1960s and the modern
implementation. For the purposes of a systems-level discussion, the most
significant difference is perhaps the massive amount of parallelism occurring
in the early 1960s design versus the modern design’s ability to perform
only one operation at a time over the microprocessor bus. For example, if
the microprocessor in Figure 1.2 is reading a word from RAM or ROM over
the bus, it cannot also be reading a word from the A/D converter at the same
time. The processor bus is a shared resource and can only support one
operation at a time.

This loss of concurrent operation arose because of the economics of
microprocessor packaging and printed-circuit board design. It’s less expen-
sive to use buses to move data into and out of packaged microprocessors
and it’s often much easier to route buses on a circuit board than to accom-
modate multiple point-to-point connections.

The consequence of creating a shared resource like a microprocessor
bus is that the use of the shared resource must be multiplexed in time. As
a result of the multiplexed operation, the operating frequency of the shared
resource must increase to accommodate the multiple uses of the shared
resource—the bus in this case. When the signal frequencies are low to start
with, as they are for a voltmeter design that’s more than 40 years old, then
the final operating frequency of the shared resource places little strain on
the system design.

However, as the various tasks performed by a system become more
ambitious causing the work to be done every clock cycle to increase, the
aggregated requirements for all of the tasks begin to approach the band-
width limit of the shared resource. As this happens, system-design mar-
gins shrink and then vanish. At that point, the system design jumps the
threshold from marginal to faulty.

As a reminder: the point-to-point architecture of the original HP 3440A
digital voltmeter concurrently operated all of the various systems blocks,
which means that there’s a lot more design margin in the interconnection
scheme than for the microprocessor-based version of the system design.
This loss of design margin is an engineering tradeoff and it undoubt-
edly reduces the implementation cost of the design, as long as no future
performance increases are envisioned that would further reduce design
margin.

1.2 FEW PINS = MASSIVE MULTIPLEXING

As shown in Figure 1.3, the 4-bit Intel 4004 microprocessor was packaged
in a 14-pin dual-inline package (DIP). Consequently, this microprocessor’s
4-bit bus not only multiplexed access to the various components in the
system, it also had to multiplex the bus-access addresses with the data on
the same four wires. It took three clock cycles to pass a 12-bit address
out over the bus and two or four more clock cycles to read back an 8- or
16-bit instruction. All instructions came from ROM in a 4004-based sys-
tem. RAM accesses were even slower because they required one instruction
to pass out the target address and then a second instruction to read data
from or write data to the selected location. With a maximum operating
frequency of 740 kHz and long, multi-cycle bus operations, the Intel 4004
microprocessor was far too slow to take on many system control tasks and the electronics design community largely ignored the world’s first microprocessor.

The world’s second commercial microprocessor, Intel’s 8008 introduced in April, 1972, was not much better than the 4004 processor in terms of bus bandwidth. The 8008 microprocessor’s 8-bit bus needed two cycles to pass a 14-bit address and one to three cycles to accept an 8- to 24-bit instruction. The 8008 microprocessor had a wider, 8-bit bus that Intel squeezed into an unconventional 18-pin DIP (shown in Figure 1.4). Although the instruction times for the 4004 and 8008 microprocessors were similar (10.5 µsec versus 12.5 to 20 µsec, respectively), the 8008 microprocessor’s RAM accesses were faster than those of the Intel 4004 processor because the Intel 8008 processor used a more conventional RAM-access cycle that output an address and then performed the data transaction during the same instruction cycle. The 8008 microprocessor ran at clock rates of 500–800 kHz. Consequently, like its predecessor, it was too slow to fire the imagination of many system designers.

**Figure 1.3**
The Intel 4004 microprocessor, introduced in 1971, was packaged in a 16-pin package that severely limited the processor’s I/O bandwidth and restricted its market accordingly. Photo Courtesy of Stephen A. Emery Jr., www.ChipScapes.com.
1.3 THIRD TIME'S A CHARM

Intel finally got it right in April, 1974 when the company introduced its third microprocessor, the 8-bit 8080. The 8080 microprocessor had a non-multiplexed bus with separate address and data lines. Its address bus was 16 bits wide, allowing a 64-Kbyte address range. The data bus was 8 bits wide. As shown in Figure 1.5, Intel used a 40-pin DIP to house the 8080 microprocessor. This larger package and the microprocessor’s faster 2-MHz clock rate finally brought bus bandwidth up to usable levels. Other microprocessor vendors such as Motorola and Zilog also introduced microprocessors in 40-pin DIPs around this time and system designers finally started to adopt the microprocessor as a key system building block.

1.4 THE MICROPROCESSOR: A UNIVERSAL SYSTEM BUILDING BLOCK

Over the next 30 years, microprocessor-based design has become the nearly universal approach to systems design. Once microprocessors had achieved
the requisite processing and I/O bandwidth needed to handle a large number of system tasks, they began to permeate system design. The reason for this development is simply engineering economics. Standard microprocessors offered as individual integrated circuits (ICs) provide a very economical way to package thousands of logic transistors in standard, testable configurations. The resulting mass-produced microprocessor ICs have become cheap, often costing less than $1 per chip, and they deliver abilities that belie their modest cost.

Microprocessors also dominate modern electronic system design because hardware is far more difficult to change than software or firmware. To change hardware, the design team must redesign and re-verify the logic, change the design of the circuit board (in pre-SOC times), and then re-run any required functional and environmental tests. Software or firmware developers can change their code, recompile, and then burn new ROMs or download the new code into the existing hardware.

In addition, a hardware designer can design a microprocessor-based system and build it before the system’s function is fully defined. Pouring the software or firmware into the hardware finalizes the design and this event can occur days, weeks, or months after the hardware has been designed, prototyped, verified, tested, manufactured, and even fielded. As a consequence, microprocessor-based system design buys the design team extra time because hardware and firmware development can occur

![FIGURE 1.5](image)

Intel finally crossed the bus-bandwidth threshold into usability by packaging its third-generation 8080 microprocessor in a 40-pin package. Many competitors swiftly followed suit (shown is Zilog’s 8-bit Z80 microprocessor) and the microprocessor quickly became a standard building block for system designers. Photo Courtesy of Stephen A. Emery Jr., www.ChipScapes.com.
concurrently, which telescopes the project schedule (at least under ideal conditions).

With the advent of practical 8-bit microprocessors in the mid-1970s, the microprocessor’s low cost and high utility snowballed and microprocessor vendors have been under great pressure to constantly increase their products’ performance as system designers think of more tasks to execute on processors. There are some obvious methods to increase a processor’s performance and processor vendors have used three of them.

The first and easiest performance-enhancing technique used was to increase the processor’s clock rate. Intel introduced the 8086 microprocessor in 1978. It ran at 10 MHz, five times the clock rate of the 8080 microprocessor introduced in 1974. Ten years later, Intel introduced the 80386 microprocessor at 25 MHz, faster by another factor of 2.5. In yet another ten years, Intel introduced the Pentium II processor at 266 MHz, better than a ten times clock-rate increase yet again. Figure 1.6 shows the dramatic rise in microprocessor clock rate over time.

Note that Intel was not the only microprocessor vendor racing to higher clock rates. At different times, Motorola and AMD have also produced...
microprocessors that vied for the “fastest clock rate” title and Digital Equipment Corporation (DEC) hand-tuned its series of Alpha microprocessors to world-beating clock rates. (That is until Compaq bought the company in 1998 and curtailed the Alpha development program. HP then acquired Compaq in late 2001.)

At the same time, microprocessor data-word widths and buses widened so that processors could move more data during each clock period. Widening the processor’s bus is the second way to increase processing speed and I/O bandwidth. Intel’s 16-bit 8086 microprocessor had a 16-bit data bus and the 32-bit 80386 microprocessor had a 32-bit data bus.

The third way to increase processor performance and bus bandwidth is to add more buses to the processor’s architecture. Intel did exactly this with the addition of a separate cache-memory bus to its Pentium II processor. The processor could simultaneously run separate bus cycles to its high-speed cache memory and to other system components attached to the processor’s main bus.

As processor buses widen and as processor architectures acquire extra buses, the microprocessor package’s pin count necessarily increases. Figure 1.7 shows how microprocessor pin count has increased over the years.

![Microprocessor Pin Count over Time](image)

**FIGURE 1.7**

Microprocessor pin counts have also risen dramatically over time due to the demand of system designers for ever more processor performance.
Like the rising curve plotted in Figure 1.6, the increasing pin count shown in Figure 1.7 is a direct result of system designers' demand for more processor performance.

1.5 THE CONSEQUENCES OF PERFORMANCE—IN THE MACRO WORLD

Faster clock rates coupled with more and wider buses do indeed increase processor performance, but at a price. Increasing the clock rate extracts a penalty in terms of power dissipation, as shown in Figure 1.8. In fact, power dissipation rises roughly with the square of the clock-rate increase, so microprocessor power dissipation and energy density have been rising exponentially for three decades. Unfortunately, the result is that the fastest
packaged processors today are bumping into the heat-dissipation limits of their packaging and cooling systems. Since their introduction in 1971, cooling design for packaged microprocessors progressed from no special cooling to:

- careful system design to exploit convection cooling,
- active air cooling without heat sinks,
- active air cooling with aluminum and then copper heat sinks,
- larger heat sinks,
- even larger heat sinks,
- dedicated fans directly attached to the processor’s heat sink,
- heat pipes,
- heat sinks incorporating active liquid cooling subsystems.

Each step up in heat capacity has increased the cost of cooling, increased the size of required power supplies and product enclosures, increased cooling noise (for fans), and decreased system reliability due to hotter chips and active cooling systems that have their own reliability issues.

SOCs cannot employ the same sort of cooling now used for PC processors. Systems that use SOCs generally lack the PC’s cooling budget. In addition, a processor core on an SOC is only a small part of the system. It cannot dominate the cost and support structure of the finished product the way a processor in a PC does. Simple economics dictate a different design approach.

In addition, SOCs are developed using an ASIC design flow, which means that gates are not individually sized to optimize speed in critical paths the same way and to the same extent that transistors in critical paths are tweaked by the designers of packaged microprocessors. Consequently, clock rates for embedded processor cores used in SOC designs have climbed modestly over the past two decades to a few hundred MHz, but SOC processors do not run at multi-GHz clock rates like their PC brethren, and probably never will because the link between smaller transistors and faster clock rates actually broke when the semiconductor industry reached the 130 nm processing node.

Gordon Moore first formulated his famous and long-lasting prediction—that shrinking transistor size and the resulting increase in device density would double roughly every two years—while working at Fairchild Semiconductor’s R&D laboratories in 1965. IBM’s Robert Dennard, the man who invented and patented the 1-transistor DRAM (dynamic RAM) in 1968, codified the linkages between physical, Moore’s law transistor scaling, increasing transistor speed, and decreasing power dissipation. He made this connection in an article titled “Design of Ion-Implanted
MOSFETs With Very Small Physical Dimensions,” which was published in the IEEE Journal of Solid-State Circuits in 1974, although Dennard and his colleagues published the basic data and conclusion two years earlier at the International Electron Devices Meeting (IEDM). The semiconductor industry has ridden this linkage between Moore’s law of transistor scaling and transistor speed and power dissipation for 30 years, until it broke. Now, transistor scaling continues to produce smaller and cheaper transistors but, with each new IC process node, the transistors are only a little faster and they no longer consume considerably less power than the immediately preceding device generation. As a result, SOC designers can no longer depend on the next semiconductor-processing node to solve performance and power-dissipation problems afflicting their system designs. To get better system performance in the 21st century, systems designers must become more sophisticated in the way they architect their systems.

1.6 INCREASING PROCESSOR PERFORMANCE IN THE MICRO WORLD

Lacking the access to the high clock rates previously available to PC processor designers, processor core designers must turn to alternative performance-enhancing strategies. Use of additional buses and wider buses are both good performance-enhancing strategies for SOC-centric processor design. In the macro world of packaged microprocessors, additional processor pins incur a real cost. Packages with higher pin counts are more expensive, they’re harder to test, and they require more costly sockets. However, in the micro world of SOC design, additional pins for wider buses essentially cost nothing. They do incur some additional routing complexity, which may or may not increase design difficulty. However, once routed, additional pins on a microprocessor core do not add much to the cost of chip manufacture, except for a fractionally larger silicon footprint.

In much the same way, additional microprocessor buses also incur very little cost penalty but provide a significant performance benefit. Processor cores for SOCs often have many more buses than their packaged counterparts, as shown in Figure 1.9.

Figure 1.9 shows a hypothetical processor core with eight buses. One of those buses, the main one, is the traditional multimaster bus that all microprocessors have had since 1971. Four of the buses communicate directly with local data and instruction memories. Two more buses communicate with instruction and data caches, respectively. The remaining bus is a fast local bus used for high-speed communications with closely coupled devices such as FIFOs and high-bandwidth peripherals.
The processor shown in Figure 1.9 has two load/store units plus an instruction-fetch unit, which can operate three of the processor's buses simultaneously. In addition, the processor's cache-control unit can independently use the cache buses and the memory buses. As a consequence, several of the buses on the processor shown in Figure 1.9 can be operating simultaneously. This sort of I/O concurrency is precisely what's needed to keep the processor core's operating frequency low, and thus keep operating power low as well. The number of pins required to implement eight buses is cost-prohibitive for a packaged processor IC, but is not at all costly for a processor core.

**FIGURE 1.9**

SOC processor cores can incorporate several buses to increase performance without incurring the pin limitations and costs of packaged processors.

The processor shown in Figure 1.9 has two load/store units plus an instruction-fetch unit, which can operate three of the processor's buses simultaneously. In addition, the processor's cache-control unit can independently use the cache buses and the memory buses. As a consequence, several of the buses on the processor shown in Figure 1.9 can be operating simultaneously. This sort of I/O concurrency is precisely what's needed to keep the processor core's operating frequency low, and thus keep operating power low as well. The number of pins required to implement eight buses is cost-prohibitive for a packaged processor IC, but is not at all costly for a processor core.

**1.7 I/O BANDWIDTH AND PROCESSOR CORE CLOCK RATE**

This dichotomy is a significant point of difference between processor chips and processor cores. A processor core's ability to support multiple simultaneous I/O transactions on several buses profoundly expands the possibilities for high-performance system architectures and topologies that would be uneconomical or impossible using packaged processor ICs for board-level system designs. Consequently, SOC designers should not
feel the same pressures to pursue processors with high clock rates that PC designers use to achieve performance goals.

However, entrenched system-design habits and rules of thumb developed from the industry’s 35 years of collective, microprocessor-based, board-level system-design experience frequently prompt even experienced system designers to compare processor cores solely on clock frequency, as they might for packaged processor chips. One of the goals of this book is to convince you that SOC processor cores are not as interchangeable as once thought and that high processor clock rates are no longer as important for SOC design as they were for high-performance, board-level system designs. In fact, as discussed above, the quest for high clock rates carries severely negative consequences with respect to power dissipation, system complexity, reliability, and cost. Achieving system performance goals at lower clock rates virtually always results in superior system designs.

Note that this does not mean that high processor-core clock rates are never important. Sometimes, just there is no alternative. However, if more task concurrency at lower clock rates can achieve system-performance goals, then that design approach will almost always prove superior. Ideally, the best SOC processor core will therefore be one that both:

- delivers high performance at low clock rates,
- can achieve high clock rates if needed.

1.8 MULTITASKING AND PROCESSOR CORE CLOCK RATE

Multitasking is another system-design choice that tends to increase processor clock rates. Processor multitasking predates the introduction of microprocessors by at least a decade. Early computers of the 1940s, 1950s, and 1960s were very expensive. Consequently, computer time was also very expensive. One way to distribute the high hardware costs was to give each computer user a share of the computer’s time—timesharing. Timeshared operating systems started to appear on computers by 1961. Multitasking is timesharing, recast. Multitasking operating systems queue multiple tasks (rather than users) and give each task a time-multiplexed share of the computer. Multitasking makes one processor appear to be doing the work of several. When computers were big and expensive, multitasking made perfect sense.

Initially, microprocessors were also expensive. The first production units of the earliest processor chips cost several hundred dollars throughout the 1970s, so there was significant financial incentive for the expensive processor to execute as many concurrent tasks as possible to amortize the processor’s cost across tasks rather than using many expensive processors to implement the multiple tasks. An entire industry has grown up around
the development of real-time operating systems for the specific purpose of making microprocessors execute multiple concurrent tasks.

Microprocessor multitasking encourages clock-rate escalation. A faster clock rate allows a processor to execute more concurrent tasks and more complex tasks. As long as processors are expensive, the system-design scales tip toward multitasking because larger power supplies and cooling components (incurred when running a processor at a higher clock rate) are probably not as expensive as a second processor. However, when processors are cheap, the scales tip against multitasking.

In 1968, a dollar bought one packaged transistor and you needed thousands of transistors to build a computer. In the 21st century, a dollar buys several million transistors on an SOC. It takes roughly 100,000 transistors to build a 32-bit RISC processor, which now cost less than a penny. Moore’s law has made transistors—and therefore processors—cheap, but conventional system-design techniques that conserve processors in exchange for increased clock rates are based on habits and rules of thumb developed when processors cost many dollars instead of less than a penny.

1.9 System-Design Evolution

Semiconductor advances achieved through the relentless application of Moore’s law have significantly influenced the evolution of system design since microprocessors became ubiquitous in the 1980s. Figure 1.10 shows how minimum microprocessor feature size has tracked Moore’s law since the introduction of the Intel 4004, which used 10-micron (10,000-nm) lithography. The figure also incorporates ITRS 2005 (International Technology Roadmap for Semiconductor) projections to the year 2020, when the minimum feature size is expected to be an incredibly tiny 14 nm. Each reduction in feature size produces a corresponding increase in the number of transistors that will fit on a chip. Presently, Intel’s dual-core Itanium-2 microprocessor holds the record for the largest number of transistors on a microprocessor chip at 1.72 billion. Most of the Itanium-2’s on-chip transistors are devoted to memory. In the 21st century, SOCs routinely contain tens of millions to several hundred million transistors.

A series of system-level snapshots in 5-year intervals illustrates how system design has changed, and how it also has clung to the past. Figure 1.11 shows a typical electronic system, circa 1985. At this point in the evolution of system design, microprocessors have been available for nearly 15 years and microprocessor-based system design is now the rule rather than the exception. Packaged microprocessor ICs are combined with standard RAM, ROM, and peripheral ICs and this collection of off-the-shelf LSI chips is arrayed on a multilayer printed-circuit board. Glue logic provides the circuits needed to make all of the standard LSI chips work together as a system.
The relentless decrease in feature size that has slavishly followed Moore's law for decades fuels rapid complexity increases in SOC designs.

By 1985, microprocessor-based system design using standard LSI parts and printed-circuit boards was common.
In this design era, the glue-logic chips are probably bipolar 74LS series TTL chips but more advanced designers are using small, field-programmable chips (usually fast, bipolar PROMs or PALs made by Monolithic Memories) for glue. Note the single-processor bus that links all of the chips together. Even though microprocessors have been available for nearly a decade and a half, the block diagram shown in Figure 1.11 could easily be representative of systems designed with Intel's original 4004 microprocessor.

Five years later, in 1990, system designers were still largely working at the board level with standard packaged microprocessor ICs. However, much of the glue logic has migrated into one or more ASICs (for high-volume systems) or FPGAs (for lower volume systems), as shown in Figure 1.12. These ASICs were usually too small (had insufficient gate count) for incorporating microprocessor cores along with the glue logic.

ASIC capacities had advanced enough to include a processor core by 1995, initiating the SOC design era. Despite the additional potential flexibility and routability afforded by on-chip system design, system block diagrams continued to look much like their board-level predecessors, as illustrated in Figure 1.13. In general, system designers used the new silicon SOC technology in much the same way they had used printed-circuit boards.
By the year 2000, increasingly advanced IC lithography permitted the incorporation of processor cores with increasingly high clock rates, which caused a mismatch between the processor's bus speed and slower peripheral devices. To uncouple the fast processor-memory subsystem from the slower peripheral sections of the SOC, system designers started to use on-chip bus hierarchies with fast and slow buses separated by a bus bridge, as shown in Figure 1.14. This system topology allows the high-speed bus to shrink in size, which reduces its capacitance and allows the processor's memory bus to keep up with the processor's high clock rates. Logically, the system block diagram of a system designed in the year 2000 still closely resembles one designed in 1985.

Present-day SOC design has started to break with the 1-processor system model that has dominated since 1971. Figure 1.15 shows a 2-processor SOC design with a control-plane processor and a data-plane processor. Each processor has its own bus and shares a peripheral device set by communicating over bus bridges to a separate peripheral bus. This arrangement is an extension of the bus hierarchy discussed in connection with Figure 1.14.

The terms “control plane” and “data plane” came into use during the Internet boom of the late 1990s and early part of the 21st century. At first, these terms referred largely to the design of multiple-board networking systems. High-speed network data passed through high-performance...
processors and hardware accelerators on a high-speed circuit board—called the data plane. Overall system control did not require such high performance so the control task was given to a general-purpose processor on a separate circuit board—called the control plane. These terms have now become universal because they suitably describe many processing systems such as video-encoding and video-decoding designs that must handle high-speed data and perform complex control.

A 2-processor design approach has also become very common in the design of voice-only mobile-telephone handsets. A general-purpose processor (almost universally an ARM RISC processor due to legacy-software and type approval considerations) handles the handset’s operating system, user interface, and protocol stack. A DSP (digital signal processor) handles the
mobile phone’s baseband and voice processing (essentially DSP functions such as Fast Fourier Transformations (FFTs) and inverse FFTs, symbol coding and decoding, filtering, etc.). The two processors likely run at different clock rates to minimize power consumption. Processing bandwidth is finely tuned to be just enough for voice processing—which minimizes product cost (mobile-phone handset designs are sensitive to product-cost differentials measured in fractions of a penny because they sell in the hundreds of millions of units per year) and also minimizes power dissipation—which maximizes battery life, talk time, and standby time.

1.10 HETEROGENEOUS- AND HOMOGENEOUS-PROCESSOR SYSTEM-DESIGN APPROACHES

Figure 1.15 shows the use of two different microprocessor cores, one general-purpose processor and one DSP. Such a system is called a
heterogeneous-multiprocessor system. A heterogeneous-multiprocessor design approach has the advantage of matching processor cores with application-appropriate features to specific on-chip tasks.

Selecting just the right processor core or tailoring the processor core to a specific task has many benefits. First, the processor need have no more abilities than required by its assigned task set. This characteristic of heterogeneous-multiprocessor system design minimizes processor gate counts by trimming unneeded features from each processor.

One of the key disadvantages of heterogeneous-multiprocessor design is the need to use a different software-development tool set (compiler, assembler, debugger, instruction-set simulator, real-time operating system, etc.) for each of the different processor cores used in the system design. Either the firmware team must become proficient in using all of the tool sets for the various processors or—more likely—the team must be split into groups, each assigned to different processor cores.

However, this situation is not always the case for heterogeneous-processor system designs, as we’ll see later in this book. Each instance of a configurable processor core can take on exactly the attributes needed for a specific set of tasks and all of the variously configured processor cores, which are based on a common ISA (instruction-set architecture), can still use the same software-development tool suite so that software team members can use familiar tools to program all of the processors in an SOC.

Some SOC designs, called homogeneous multiprocessor systems, use multiple copies of the same processor core. This design approach can simplify software development because all of the on-chip processors can be programmed with one common set of development tools. However, processor cores are not all created equally able. General-purpose processors are not generally good at DSP applications because they lack critical execution units and memory-access modes. A high-speed multiplier/accumulator (MAC) is essential to efficient execution of many DSP algorithms but MACs require a relatively large number of gates so few general-purpose processors have them.

Similarly, the performance of many DSP algorithms can benefit from a processor’s ability to fetch two data words from memory simultaneously, a feature often called XY memory addressing. Few general-purpose processors have XY memory addressing because the feature requires the equivalent of two load/store units and most general-purpose processors have only one such unit.

Although basic, voice-only mobile-telephone handsets generally have only two processors, incorporation of multimedia features (music, still image, and video) has placed additional processing demands on handset system designs and the finely tuned, cost-minimized 2-processor system designs for voice-only phones simply lack processing bandwidth for these additional functions. Consequently, the most recent handset designs with new multimedia features are adding either hardware-acceleration blocks or “application processors” to handle the processing required by the additional
features. The design of multiple-processor SOC systems is now sufficiently common to prompt a new term that describes the system-design style that employs more than one processor; multiple-processor SOCs are called MPSOCs. Figure 1.16 is a die layout of one such device, the MW301 media processor for camcorders and other video and still-image devices. The MW301 incorporates five processor cores. Four identical cores share the work of MPEG4 video coding and decoding. A fifth core handles the audio processing.

The MediaWorks' MW301 media processor is a good example of an SOC that transcends the special-purpose nature of early SOC designs. With five firmware-programmable processors, the MW301 is flexible enough to serve many applications in the audio/video realm. Consequently, an SOC like the MW301 media processor is often called a “platform” because the SOC's hardware design can be used for a variety of applications and its function can be altered, sometimes substantially, simply by changing the code it runs. There's no need to redesign the SOC hardware for new applications.

Platform SOCs are far more economical to design because they sell in larger volumes due to their flexible nature. Design and NRE (non-recurring engineering) costs can therefore be amortized across a larger chip volume, which reduces the design and NRE burden placed on the total cost of each chip sold.

The current record holder for the maximum number of microprocessor cores placed on one chip is Cisco's SPP (silicon packet processor) SOC, designed for the company's CRS-1 92-Tbit/sec router. The massively parallel SPP network processor chip incorporates 192 packet-processing elements (PPEs) organized in 16 clusters. Each cluster contains 12 PPEs and each PPE incorporates one of Tensilica's 32-bit Xtensa RISC processor cores, a very small instruction cache, a small data memory, and a DMA (direct memory access) controller for moving packet data into and out of the PPE's data memory. Figure 1.17 shows a diagram of the SPP network processor's design.

Cisco's SPP chip measures 18 mm on a side and is manufactured in a 130 nm process technology by IBM. Although there are 192 PPEs on the SPP die, four of the processors are spares used for silicon yield enhancement so the SPPs operate as 188-processor devices when fielded. There are approximately 18 million gates and 8 Mbits of SRAM on the SPP chip. Each of the PPEs on the chip consumes about 0.5 mm².

Cisco's earlier router chips employed hard-wired logic blocks to implement the router algorithms. Because of the number of features Cisco wanted to build into the CRS-1 router, and because networking standards change continuously, it was clear that the CRS-1 architecture required a firmware-programmable architecture. This requirement was brought into sharp focus as the effort to migrate from Internet protocol IPv4 to IPv6 escalated. The firmware-programmable SPP architecture allowed the CRS-1 design team to accommodate this major change in the protocol specification.
The MediaWorks MW301 media processor chip is a DVD resolution MPEG video and audio encoder/decoder system on a chip for solid state camcorder and portable video products. The chip contains a set of five loosely coupled heterogeneous processors. This is the highest performance fully C-code programmable media processor ever built. Photo courtesy of MediaWorks.
1.11 THE RISE OF MPSOC DESIGN

MediaWorks’ MW301 and Cisco’s SPP are noteworthy examples of 21st-century MPSOC design. Both system designs employ processor cores extensively, preferring them to large blocks of manually designed, custom RTL (register transfer level). This preference reflects a change in SOC-design philosophy from the design style in vogue just a few years before. In the late 1990s, most SOC designs employed just one processor core. Any additional processing was performed by RTL blocks. The reason for using this older design style was, primarily, a scarcity of gates on chip.
ASIC technology at the time could scarcely accommodate one processor core and its memory. Squeezing several processors onto a chip was simply out of the question.

Consequently, any additional processing to be performed had to be assigned to custom-designed logic. When there were few such blocks of custom logic, chip verification wasn’t too difficult. However, with climbing gate counts driven by Moore’s law, the number of gates available to the system design has rocketed into the tens of millions. Consequently, chip-level verification has become a burden—onerous in fact. Verification now consumes approximately 70% of an SOC project’s development time and budget. This design style, with its reliance on large numbers of manually designed custom-logic blocks, is not sustainable in the 21st century.

Whenever system-design styles are forced to change by technological advances, a few early adopters jump on the bandwagon. However, many design engineers remain skeptical of the need to change even when confronted by both the need and the historical facts surrounding previous changes. In the 1980s, for example, systems and chips were designed using schematics. Synopsys and its competitors started to sell logic-synthesis tools in the latter part of that decade, but it was an uphill battle for years. Most chip designers would not abandon their existing design styles, which were based on schematic capture. By the late 1990s, 10 years later, the schematic-capture design style for digital IC design had essentially disappeared and all SOCs are now designed using hardware description languages—either Verilog or VHDL.

In 1990, very few ASICs incorporated even one processor core. There simply weren’t enough gates available on economically sized chips to make on-chip processors practical. ASICs designed during the early 1990s primarily served as glue-logic chips that connected packaged processor ICs to memories and peripheral devices, as discussed previously. Ten years later, at the turn of the century, nearly every ASIC incorporated at least one on-chip processor core because the semiconductor technology permitted it and because it was more efficient and less expensive for the processor to reside on the chip with other system components.

By 1995, RISC microprocessor cores were just starting to emerge as ASIC building blocks because their compact nature (relatively low gate count) gave them a very attractive capability-per-gate quotient and their programmability provided useful design flexibility. As semiconductor technology and design styles evolved over the next 10 years, processor cores became sufficiently pervasive in ASIC design to give rise to a new name, the system on chip or SOC. Of course, once systems designers started using one on-chip processor core, it was only a matter of time before they started using two, three, or more processor cores to achieve processing goals. The high-water mark to date for Tensilica’s customers is 192 processor cores.

Despite numerous dire predictions over the past several decades, Moore’s law has not stopped nor will it for many years. Semiconductor process
technology will continue to advance. As it does, the increasing practicality of SOC designs like the MediaWorks MW301 and Cisco's SPP will cause all system designers to adopt multiple-processor system design. This change is inevitable because the semiconductor technologies, tool flows, and existing technical skills of working design engineers all support it. In fact, it is so predictable that the rapid rise in the number of on-chip processors is now built into the ITRS semiconductor roadmap.

### 1.12 VEERING AWAY FROM PROCESSOR MULTITASKING IN SOC DESIGN

The contemporary design trend toward increasing the number of on-chip processor cores is a significant counter trend to decades-old multitasking. Where multitasking loads multiple tasks onto one processor—which increases software complexity, forces processor clock rates up, and therefore increases processor power dissipation—use of multiple on-chip processors takes system design in a different direction by reducing the number of tasks each processor must execute. This design style simplifies the software by reducing the possibility of intertask interference, cutting software overhead (it takes extra processor cycles to just schedule and track multiple software tasks), and thus moderating the rise of processor clock rates.

Multitasking was developed when microprocessors were expensive; when every processor cycle was precious; and when the use of multiple processors was completely out of the question for reasons of engineering economics, circuit-board real estate, and power dissipation. However, Moore's law has now reduced the cost of silicon for an on-chip processor to mere pennies (or less) and these costs will further decline as the relentless application of Moore's law continues to advance the semiconductor industry's ability to fit more transistors on a chip. Microprocessor cores on SOCs are no longer expensive—and they get cheaper every year. All system-design techniques based on the old assumptions about processor costs must be rethought just as system-design techniques had to be rethought when logic-synthesis tools started to appear. Decades old, pre-SOC system-design techniques that conserve processors (which are now cheap) in the face of increasing power dissipation, software complexity, and development cost are clearly obsolete in the 21st century.

### 1.13 PROCESSORS: THE ORIGINAL, REUSABLE DESIGN BLOCK

Microprocessors became successful because they were the first truly universal, reusable block of logic to become available. With firmware
reprogramming, microprocessors could be made to perform a very wide range of tasks with no changes to the hardware. This characteristic allowed system designers to use fixed-ISA, packaged processor ICs in an ever-expanding number of systems. As the popularity of these universal system building blocks grew, an entire software-development tool industry grew up around packaged microprocessors. Large numbers of compiler and RTOS (real-time operating system) vendors popped into existence in the 1980s, the decade when microprocessors became firmly established in the system designers’ lexicon.

When systems design began to migrate from the board level to the chip level, it was a natural and logical step to continue using fixed-ISA processor cores in SOCs. Packaged processors had to employ fixed ISAs to achieve economies of scale in the fabrication process. System designers became versed in the selection and use of fixed-ISA processors and the related tool sets for their system designs. Thus, when looking for a processor to use in an SOC design, system designers first turned to fixed-ISA processor cores. RISC microprocessor cores based on processors that had been designed for personal computers and workstations from ARM and MIPS Technologies were early favorites due to their low gate count.

However, when you’re designing custom silicon, there’s no technical need to limit a design to fixed-ISA microprocessor cores as there is for board-level systems based on discrete, pre-packaged microprocessors. If there’s legacy software to reuse, there’s certainly a reason to retain a particular microprocessor ISA from one system design to the next. However, if there is no legacy code or if the legacy code is written in C, system designers have a freer hand in the selection of a processor with a different ISA if such a processor improves the system’s performance, power dissipation, or manufacturing cost.

The first practical configurable microprocessor cores started to appear in the late 1990s. A configurable processor core allows the system designer to custom tailor a microprocessor to more closely fit the intended application (or set of applications) on the SOC. A “closer fit” means that the processor’s register set is sized appropriately for the intended task and that the processor’s instructions also closely fit the intended task. For example, a processor tailored for digital audio applications may need a set of 24-bit registers for the audio data and a set of specialized instructions that operate on 24-bit audio data using a minimum number of clock cycles.

Processor tailoring offers several benefits. Tailored instructions perform assigned tasks in fewer clock cycles. For real-time applications such as audio processing, the reduction in clock cycles directly lowers operating clock rates, which in turn cuts power dissipation. Lower power dissipation extends battery life for portable systems and reduces the system costs associated with cooling in all systems. Lower clock rates also allow the SOC to be fabricated in slower and therefore less expensive IC-fabrication technologies.
Even though the technological barriers to freer ISA selection were torn down by the migration of systems to chip-level design, system-design habits are hard things to break. Many system designers who are well versed in comparing and evaluating fixed-ISA processors from various vendors elect to stay with the familiar, which is perceived as a conservative design approach. When faced with designing next-generation systems, these designers immediately start looking for processors with higher clock rates that are just fast enough to meet the new system’s performance requirements. Then they start to worry about finding batteries or power supplies with extra capacity to handle the higher power dissipation that accompanies operating these processors at higher frequencies. They also start to worry about finding ways to remove the extra waste heat from the system package. In short, the design approach cited above is not nearly as conservative as it is perceived; it is merely old fashioned. In Valley Speak (that is, San Fernando Valley), “It is so last century.”

It is not necessary for SOC designers to incur these system-design problems if they use the full breadth of the technologies available instead of limiting themselves to older design techniques developed when on-chip transistors were not so plentiful. Moore’s law has provided new ways to deal with the challenges of rising system complexity, market uncertainties, and escalating performance goals. This book focuses on one such system technology: configurable and preconfigured, application-specific processor cores.

1.14 A CLOSER LOOK AT 21st-CENTURY PROCESSOR CORES FOR SOC DESIGN

Microprocessor cores used for SOC design are the direct descendents of Intel’s original 4004 microprocessor. They are all software-driven, stored-program machines with bus interconnections. Just as packaged microprocessor ICs vary widely in their attributes, so do microprocessors packaged as IP cores. Microprocessor cores vary in architecture, word width, performance characteristics, number and width of buses, cache interfaces, local memory interfaces, and so on. Early on, when transistors were somewhat scarce, many SOC designers used 8-bit microprocessor cores to save silicon real estate. In the 21st century however, some high-performance 32-bit RISC microprocessor cores consume less than 0.5 mm² of silicon, so there’s no longer much reason to stay with lower performance processors. Indeed, the vast majority of SOC designers now use 32-bit processor cores.

In addition, microprocessor cores available as IP have become specialized just like their packaged IC brethren. Thus you’ll find 32-bit, general-purpose processor cores and DSP cores. Some vendors offer other sorts of
very specialized microprocessor IP such as security and encryption processors, media processors, and network processors. This architectural diversity is accompanied by substantial variation in software-development tools, which greatly complicates the lives of developers on SOC firmware-development teams.

The reason for this complication is largely historic. As microprocessors evolved, a parallel evolution in software-development tools also occurred. A split between the processor developers and tool developers opened and grew. Processor developers preferred to focus on hardware architectural advances and tool developers focused on compiler advancements. Processor developers would labor to produce the next great processor architecture and, after the processor was introduced, software-tool developers would find ways to exploit the new architectural hardware to produce more efficient compilers.

In one way, this split was very good for the industry. It put a larger number of people to work on the parallel problems of processor architecture and compiler efficiency. As a result, it's likely that microprocessors and software tools evolved more quickly than if the developments had remained more closely linked.

However, this split has also produced a particular style of system design that is now limiting the industry's ability to design advanced systems. SOC designers compare and select processor cores the way they previously compared and selected packaged microprocessor ICs. They look at classic, time-proven figures of merit such as clock rate, main-bus bandwidth, cache-memory performance attributes, and the number of available third-party software-development tools to compare and select processors. Once a processor has been selected, the SOC development team then chooses the best compiler for that processor, based on other figures of merit. Often, the most familiar compiler is the winner because learning a new software-development tool suite consumes precious time more economically spent on actual development work.

If the SOC requires more than one processor, it's often because there's specialized processing to be done. In the vast majority of such cases, there is some sort of signal or image processing to be performed and a general-purpose processor isn't an efficient choice for such work. Usually, this situation leads to another processor selection, this time for a DSP and an associated software-development tool suite.

The big problem with this selection method is that it assumes that the laws of the microprocessor universe have remained unchanged for decades. This assumption is most definitely flawed in the 21st century. Processor cores for SOC designs can be far more plastic than packaged microprocessor ICs for board-level system designs. Shaping these processor cores for specific applications produces much better processor efficiency and much lower system clock rates. In short, using the full abilities of microprocessor core IP to more closely fit the application problems produces better system designs.
Certainly, many microprocessor cores are not at all plastic. They lack configurability. However, a relatively new class of microprocessor core, the configurable microprocessor, exploits the plastic nature of the SOC's underlying silicon to permit the creation of processors that perform much better in SOC designs than older microprocessor cores, which are based on processor architectures that were originally developed to fit into DIPs and thus inherit some of the limitations placed on packaged microprocessor ICs.

Tensilica's Xtensa processor family is an example of such a configurable core. The Xtensa architecture was specifically designed for SOC use. It was designed to fully exploit the plastic nature of nanometer silicon. The original Xtensa processors were available as fully configurable processor cores. These processor cores have now been joined by a family of preconfigured microprocessor cores called the Diamond Standard Series, which is based on and compatible with Xtensa processors. Members of the Diamond Standard Series of processor cores have already been tailored for specific SOC applications and are not further configurable.

Together, the Xtensa and Diamond processor cores constitute a family of software-compatible microprocessors covering an extremely wide performance range from simple control processors, to DSPs, to 3-way superscalar processors. The configurability of Xtensa processors allows the performance range to grow even further, to the limits supported by the underlying silicon. Yet all of these processors use the same software-development tools so that programmers familiar with one processor in the family can easily switch to another.

As a consequence of this microprocessor core family's wide performance range, it's entirely possible to develop entire SOCs using only Xtensa and Diamond microprocessor cores, barring issues of legacy software. In fact many SOCs have already been designed in just this manner. The benefit of this system-design approach is that it boosts both hardware- and firmware-development productivity in a way that cannot be approached when using old system-design methods that employ different processor architectures for different tasks on the SOC—and therefore different processor-interface schemes and different software-development tools.

This book emphasizes a processor-centric MPSOC design style shaped by the realities of the 21st-century and nanometer silicon. It advocates the assignment of tasks to firmware-controlled processors whenever possible to maximize SOC flexibility, cut power dissipation, reduce the size and number of hand-built logic blocks, shrink the associated hardware-verification effort, and minimize the overall design risk. The design examples in this book employ members of the Xtensa and Diamond Standard processor families because of the extremely broad performance range and extended I/O abilities that these processor cores offer as a group. The advanced SOC design styles discussed in this book can be used with other microprocessor cores to create MPSOCs, but it will be more difficult to achieve the same results.

