# Effect of increasing chip density on the evolution of computer architectures

by R. Nair

**Trends in lithography and process technology indicate that billion-transistor computer chips will be possible well before the end of the decade. Such a large number of transistors could be used to implement dynamic learning techniques to improve the performance of a processor for many applications. However, the efficiency of use of transistors in this manner is not high. A more attractive use of the available transistors is to bring more of the entire system onto the chip, and this paper examines two different approaches for doing so. The first involves bringing memory closer to the processors in a symmetric multiprocessor cell, and using these cells in a regular organization with a programmable interconnection to create powerful computers. The second involves integration on the same chip of varied structures such as processors, DRAM, sensors, and transducers, which in the past required different processing capabilities—commonly referred to as the System-on-a-Chip approach. The paper describes the exciting options offered by both approaches and discusses the implications of each for programming and tool development.**

## Introduction

Since the early 1970s, we have witnessed a relentless drive toward smaller features and hence higher functionality on semiconductor chips. Despite predictions to the contrary, this trend does not appear to be letting up. In fact, in the past few years, competition among semiconductor manufacturers has actually led to even higher chip functionality than one would expect from simple extrapolation of trend lines. Today, in 2001, high-performance processor chips such as the POWER4 [1] contain more than 170 million transistors. Since it is a matter of a few years before billion-transistor chips become commonplace, it is not too early to ask, "What functions will be expected of billion-transistor chips, and how will they be organized?" Any answer to this question can at best be an educated guess—innovation and changing market forces often produce surprises.

The Semiconductor Industry Association (SIA) has coordinated the efforts aimed at projecting what processing technology will be capable of providing in the



0018-8646/02/$5.00 © 2002 IBM

**Table 1** Summary of projections from the SIA roadmap. High-performance designs use more custom design techniques to pack more function with better performance in a given space.

| | | | *1999* | *2002* | *2005* | *2008* | *2011* |
|---|---|---|---|---|---|---|---|
| Technology | | nm | 180 | 130 | 100 | 70 | 50 |
| Gate length | | nm | 140 | 85–90 | 65 | 45 | 30–32 |
| Density | DRAM | $Gb/cm^2$ | 0.27 | 0.71 | 1.63 | 4.03 | 9.94 |
| | SRAM | Million transistors per $cm^2$ | 35 | 95 | 234 | 577 | 1423 |
| | High-performance logic | | 24 | 65 | 142 | 350 | 863 |
| | ASIC logic | | 20 | 54 | 133 | 328 | 811 |
| | High-volume logic | | 7 | 18 | 41 | 100 | 247 |
| Local clock frequency | High-performance | GHz | 1.25 | 2.1 | 3.5 | 6.0 | 10.0 |
| | ASIC | | 0.5 | 0.7 | 0.9 | 1.2 | 1.5 |
| | High-volume | | 0.6 | 0.8 | 1.1 | 1.4 | 1.8 |

future. Its latest document, released in 1999 [2] as the *International Technology Roadmap for Semiconductors* (ITRS), provides some insight into chip and package characteristics until 2014. Projections provided by previous roadmaps have been accurate. These roadmaps have been widely used for planning not only by semiconductor manufacturers but also by suppliers of equipment and materials, and the coordinated effort to meet these goals has therefore made the SIA projections self-fulfilling. A summary of the ITRS projections appears in **Table 1**. The reader is referred to the introduction in the roadmap for details on interpreting the data in the table. According to the table, the pace of device miniaturization is projected to continue unabated for at least another decade. A graphical view of this is presented in **Figure 1**, which shows the expected number of transistors on various types of chips assuming a constant die size of 400 $mm^2$. If events unfold as suggested in the roadmap, a billion-transistor processor chip is a distinct possibility before 2008. In fact, it could be sooner, if larger die sizes (up to 800 $mm^2$) as projected in the ITRS become possible.

The uses for such a large number of transistors are varied. Computer architects have conflicting views on how they would organize these transistors on a chip. Until now, designers of high-performance CMOS processors have used the increasing chip real estate principally to improve the performance of a uniprocessor core, either through sophisticated microarchitecture techniques or by adding SRAM caches. Some scientists feel that this trend will continue; they argue that as long as the performance of uniprocessors can be improved, it is worth doing so, because systems employing multiprocessors presumably can always benefit from faster uniprocessors. This paper argues that there are diminishing returns from adding transistors to uniprocessors and that there are other attractive uses for high-density technology of the future.

## The billion-transistor uniprocessor

In a special issue of *Computer* magazine [3], three sets of researchers postulated that the best use of a billion transistors on a chip was to incorporate mechanisms that would improve the performance of a single thread of computation. Each paper had a different opinion on how this could be done, but all of them used techniques based on gathering information from the past behavior of a computational thread to predict its future behavior. Below is a sample of the opinions expressed:

- Patt et al. [4] postulate that for better parallel exploitation of large numbers of functional units, it is necessary to predict the stream of instructions to be executed. This implies large numbers of transistors both in prediction hardware and in large caches needed to feed the parallel functional units.
- Lipasti et al. [5] also devote a large fraction of the chip to various types of prediction, especially data value prediction and memory alias prediction. Such prediction allows the processor to utilize parallel functional units better by speculating on register and memory dependence.
- Smith et al. [6] express concern about communication delays between functional units and register files on a chip. They suggest that future uniprocessors will consist of several simple and fast pipelines interconnected loosely, with one central unit examining the thread to be executed and supplying the pipelines with predicted streams of instructions, and another unit supplying predicted or actual data to feed these pipelines.

**Figure 2** shows that instruction processing has graduated from the serial processors of the 1940s to the pipelined processors of the 1960s and later to the superscalar processors of the 1980s [6]. The performance benefit gained in progressing from one generation to the next has been at the cost of transistors needed to support concurrent operations. In each transition, the proportion of transistors essential to the processing of each instruction has become a smaller fraction of the total number of transistors deployed on the processor—many additional transistors are needed to ensure that the added parallel units are kept busy. Moreover, the performance benefit in each transition is not uniform across all applications, since some applications exploit additional hardware better than others. The efficiency of use of transistors in a processor has therefore been decreasing steadily and, as explained later, many performance-boosting techniques being proposed today may actually end up degrading the performance of some applications outside the domain for which they were intended.

Olukotun et al. [7] compared a MIPS[1]**-like processor chip having 32K instruction and data caches and capable of issuing six instructions in parallel with a two-issue processor and 8K caches. They showed that the area of the simpler processor was less than a quarter of the area of the wider processor, and yet the performance advantage of the wider machine was less than a factor of 1.6 for most applications. In essence, Amdahl's law [8] is at play even in instruction-level parallelism. In order to improve the performance of wide superscalar machines, it is necessary to increase the performance of the memory subsystem and of the branch-prediction mechanism. Simply increasing the sizes of caches provides no return in small applications (whose working set fits in small caches) and diminishing returns in large applications, especially many commercial and scientific applications characterized by streaming behavior of data. Similarly, some important commercial applications actually benefit from simple predictors because of their lower learning overhead [9]. Applications with large code footprints require large tables, rather than elaborate mechanisms, to improve their predictability. However, large tables, like large caches, are expensive in area as well as in access time and may degrade the performance of scores of applications that do not need them.

Would the situation be better if the level of parallelism were increased, simply by doing more of what was done in the transition from the first generation to the second, i.e., by increasing the clock frequency through increased pipelining? The graph of **Figure 3** shows results of simulating the effect of increasing the number of pipeline

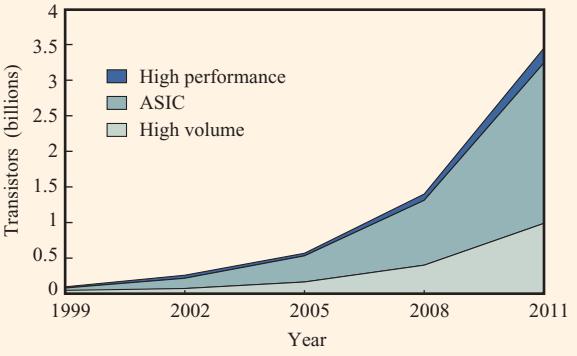


**Figure 1**

Number of transistors on various types of chip (400 mm$^2$) as projected by the SIA roadmap.

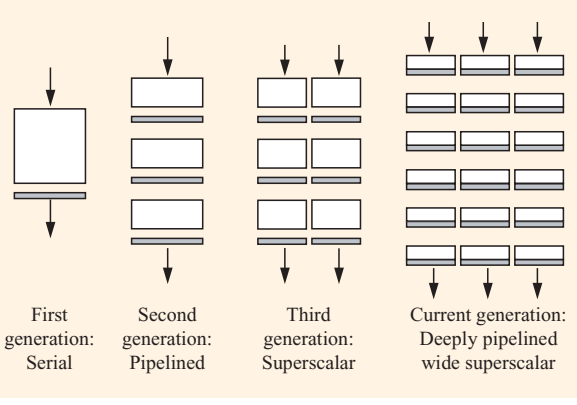


**Figure 2**

Illustration of increasing parallelism implemented in successive processor generations. Adapted with permission from [6]; © 1997 IEEE.

stages of a fictitious out-of-order[2] PowerPC* processor with small issue width. Two benchmarks were simulated: One was the *go* program from the SPECint95** benchmark suite, and the other was a transaction-processing benchmark on DB2*. It is clear from the figure that there is little to be gained from increasing the frequency for the transaction-processing benchmark. In fact, the complexities associated with increasing the clock rate, and the likely increase in power dissipated, argue that it may be detrimental to increase the pipelining for workloads of this type. On the other hand, the asymptotic

[1] Instruction set architecture of the MIPS Computer Systems, now owned by Silicon Graphics, Inc.

[2] An *out-of-order* processor is one that allows the execution of instructions in an order different from the one originally specified, but maintains the appearance to the program of having completed the instructions in original program order.

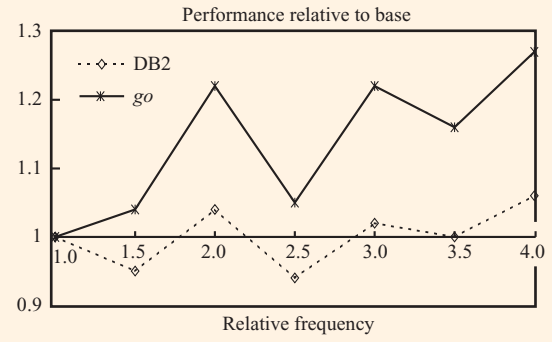


**Figure 3**

Effect of increasing pipeline depth for two applications: a transaction-processing application running on DB2 and the *go* benchmark from the SPECint95 suite.

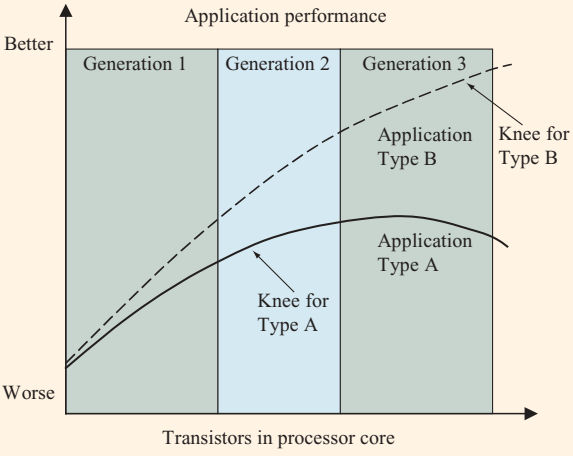


**Figure 4**

Processor-application efficiency curve.

performance of the *go* benchmark does improve with frequency, but increasing the frequency fourfold provides less than 30% improvement. Once again, the area overhead due to deeper pipelining, increased costs in design and verification, increased design time, and increased power dissipation could act to counter this benefit. (The local dips in both graphs exist because it is not possible to simply add pipeline stages to an existing design to achieve an arbitrary frequency; there are limited ways of partitioning a piece of combinational logic in front of a pipeline latch, and even these may involve duplication of logic and hence further area overhead.)

The foregoing observations suggest that the first-to-second-generation transition was successful because most uniprocessor applications responded well to pipelining (and hence the increased investment in transistors). Fewer applications continued to respond well in the transition to the third generation, and fewer still will keep up with further increases in processor complexity. **Figure 4** shows the general performance[3] characteristics of two applications: Type A is an application which has large instruction and data working sets, is not easily predictable, and relies on system-provided libraries and functions; Type B is a well-behaved scientific application which can be restructured and recompiled to take advantage of the underlying implementation. This graph, which we refer to as the "efficiency curve," shows a flattening of performance with increasing complexity; at some point, the overhead of a large processor could actually cause a performance degradation. There is a "knee" to the efficiency curve for most applications.[4] The location of this knee depends on the nature of the application—beyond the knee, the processor is not cost-effective for the application. Robustness in performance across application domains is an important characteristic of general-purpose processors; in this respect, many applications running on today's high-performance processors are already well past the knee of the efficiency curve. In some cases, it may be possible to restructure the algorithm for an application to shift its knee toward the right. But ironically, the very success of high-performance processors and the drive toward ease in creating new applications has led to the proliferation of new applications and subsystems, such as object-oriented programming, whose large code and data footprints and relatively low predictability put them closer to Type A in Figure 4. With the knee of the efficiency curve of general workload shifting to the left in this manner, it will only become harder to justify a billion-transistor uniprocessor chip.

## Multiple simpler processors on a chip

In many commercial applications, the biggest detractor to good performance is the response time (latency) of access to main memory. Over the years this latency, as measured in number of processor cycles, has increased from a few cycles to several hundred cycles. One reason is that pipelining techniques have helped reduce the cycle time of processors dramatically. Another important reason is the increased understanding of the strengths of shared-memory multiprocessing (SMP) [10], and the growing exploitation of SMP in commercial systems. Multithreaded applications are more common today than they have ever

[3] The *performance* of an application is the reciprocal of the time taken to complete execution of the application on the processor.
[4] If the efficiency curve asymptotically reached a maximum value, one could define the "knee" as the number of transistors for which the attained performance is some fixed percentage, say 90%, of this maximum value.

been, and there is increasing emphasis in the commercial world on increased throughput of multithreaded applications rather than the latency of a single thread. However, when multiple processors share the same address space, a larger physical memory is needed to reduce the effect of contentions between processors. A larger memory, especially one that is accessed through a high-bandwidth interconnection network, adds latency to loads and stores from each processor and reduces utilization of processor resources.

In many applications, adding transistors to improve the performance of each uniprocessor cannot offset the performance loss due to increased memory latency. A better solution for these applications is to reduce the latency and increase the bandwidth to memory by bringing memory and processors together on a die. Until now, technology has not allowed a reasonably sized multiprocessor on a die. By 2008, however, a 400-$mm^2$ chip would easily be able to accommodate 16 processors running at 6 GHz or higher. If designed for optimal efficiency for typical applications, each processor should take up only about 5 $mm^2$ on the die,[5] leaving approximately 320 $mm^2$ for memory and communication. As depicted in **Figure 5**, a part of this area will be taken up by the interconnection network and other logic to keep caches coherent and to maintain access consistency. Allowing for 50 $mm^2$, we still have more than half the chip—270 $mm^2$—remaining for memory. At a projected SRAM density of 1423 million transistors per $cm^2$, this allows for roughly 64 MB of SRAM. On the other hand, at 4 $Gb/cm^2$ one could put 11 Gb of DRAM in the same area, assuming that the DRAM-with-logic technology matures as expected. By 2011, according to Table 1, chip densities would have improved to a point at which the processors and switch in such a 16-way SMP running at 10 GHz would occupy only 50 $mm^2$, and the rest of a 400-$mm^2$ chip could accommodate 35 Gb of DRAM. The larger DRAM provides some important advantages over SRAM—it has lower power dissipation per unit area, it reduces off-chip misses significantly for many commercial applications, and it allows the chip to be used as a standalone multiprocessor.

While SMP is a convenient paradigm for programming, it does have an unfortunate limitation: scalability. As the number of processors in an SMP system increases, the complexity of the interconnection network between the processors and memory increases, and consequently the cost of these systems increases. Simpler interconnection networks are less effective and can easily become saturated because the probability increases that the valid contents of a requested memory location is in the cache of another processor and causes traffic over and above the usual traffic between the processors and memory. As a result, many applications actually degrade in performance with large numbers of processors [11]. The cost-effective number of processors (i.e., the number beyond which there is no significant performance gain for additional processors) varies with the application.

[5] This area could accommodate a two-issue out-of-order processor with 32K instruction and data caches and a single-instruction-multiple-data (SIMD) multimedia unit.

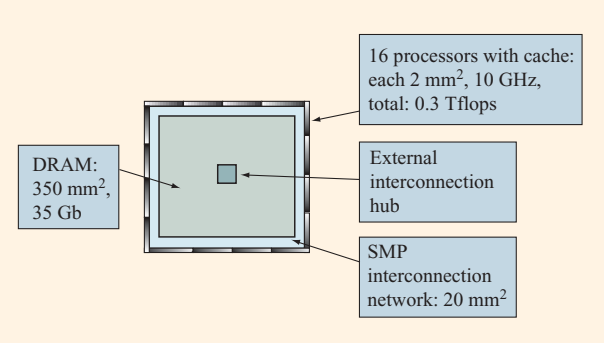


**Figure 5**

A 35Gb DRAM chip with 16 processors forming an SMP-on-a-chip—possible with technology projected for 2011.

It is usually the case that the few applications which do well on large SMPs tend to have low communication overhead and typically can be rewritten to be exploited by less tightly coupled distributed systems. With appropriate structuring, these applications could be made to work well also with a clustered system [12] built with a relatively loose interconnection of small SMP nodes. A 16-way SMP with on-chip DRAM would be an ideal node, for example, for a multi-node clustered system such as the IBM S/390* Parallel Sysplex* [13].

We have suggested reasons why an appropriate configuration for general-purpose processor chips of the future may be a small SMP (e.g., 16-way) with a fair amount of DRAM on chip, rather than a large single-chip uniprocessor. The IBM POWER4 [1] chip is a step in that direction. It has eschewed the use of high integration density for obtaining maximum uniprocessor performance, instead settling for greater throughput per unit area by putting two cores and a large SRAM Level 2 cache on a single chip.

## Migrating from the Von Neumann style of computing

The availability of a reasonably sized SMP with a large amount of memory on a single chip could be an important turning point in the computation world. The separation of an expensive computing element from an inexpensive memory element is a heritage of the original Von Neumann architecture [14]. It has proven highly successful over

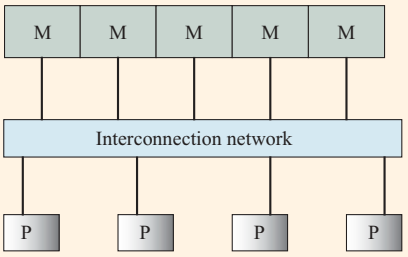


Figure 6

The dance-hall structure of shared-memory symmetric multiprocessing systems (P: processor with caches; M: memory node).

the years, spurred by its use in popular programming languages and environments. The Von Neumann model provides a balanced machine in an economic sense as long as the cost-to-utilization ratio of the computing structure is similar to that of the memory structure. However, complex superscalar processors of today are becoming more costly to develop, and their speculative nature causes them to be utilized less effectively than ever before. The computing element is declining in its effectiveness both because of inefficient use of transistors and because of the ever-increasing latency to main memory. Techniques to hide this latency, such as caches, which worked well when latencies were lower, are becoming less effective, and processors spend a lot of time waiting for data to move back and forth from memory.

The advent of multiprocessing was considered an opportunity to change the computing landscape. Proposals were made and machines were built that promoted greater use of memory by dedicating smaller portions of memory to simple, inexpensive computing engines. However, even though a large number of interesting problems could be solved effectively using these machines, economies of scale made the cost of developing them large in comparison with the cost of developing the popular Von Neumann-style engines. The low cost of the latter also led to the popularity of the "dance-hall" paradigm of memory on one side and processing on the other, as in the symmetric multiprocessor (SMP) paradigm shown in **Figure 6**. Over decades the combined investment in these forms of computing will make it difficult to dislodge either the Von Neumann model at the uniprocessor level or the SMP model, its derivative at the multiprocessor level. It appears that large systems of the future will have to build on past investment in processor design and software, and exploit the low cost of popular systems in order to venture into a world beyond Von Neumann computing. One such possibility is examined in the next section.

### *Cellular architectures*

Over the last three decades, scientists have become adept at exploiting parallelism at various granularities, especially for scientific applications. It is often possible for many important problems to exploit parallelism simultaneously at multiple levels, starting at the instruction level, through the multiprocessor level, up to the cluster level, typically by partitioning a problem into several threads of computation with little communication overhead between them. This has led to several interesting architectures to solve these classes of problems.

One such architecture was that proposed for the Connection Machine [15], which attempted to match thousands of processing elements to the natural structure of elements in a problem. The machine had built-in connections between processing elements, but the optimal interconnection needed between the processing elements was configured directly by the application software because the interconnection required invariably depends on the problem being solved. The first design requirement in a connectionist architecture, as we refer to this style, is a large number of processing elements, each element at any given time dedicated to the computation associated with a basic element in the problem to be solved. Such an approach often localizes the communication requirements and reduces the required memory-to-computation ratio.

A similar philosophy is taken by the IBM Blue Gene* machine [16] currently under development. The Blue Gene machine is being designed with the requirements of the protein-folding problem in mind. This problem also appears to enjoy a relatively low memory requirement per computation thread. Each chip in the Blue Gene system (150-nm technology) is expected to have 8 MB of DRAM and aims to get a peak performance of 32 gigaflops using 32 floating-point engines. A system comprising 32 000 such chips can potentially provide 1 petaflops of computing power.

The connectionist architecture, characterized by large numbers of processors but little memory per processor, is appealing for solving many simulation problems in which behavior in the large (i.e., of the system as a whole) can be completely characterized by behavior in the small (i.e., at the small element level). However, many simulation problems benefit from a more hierarchical treatment; at some level, it is adequate and expedient to model an ensemble behavior, while at another level it is necessary to perform calculations on local interactions of smaller elements. Simulation of the ensemble behavior often involves unstructured computation and a comparatively higher memory-to-computation ratio. The connectionist architecture is better suited to solve problems at a finer granularity than at the ensemble level.

A similar hierarchical-granularity situation is also evident in nonsimulation problems, especially design

problems in which the design space is too large to perform exhaustive simulations leading to an optimal solution. One example of such a problem is that of routing wires in integrated circuit chips [17]. While it is possible to model this problem by simulating each intersection in the wiring grid as a cell, a more effective way to perform this task is to first do a global wiring at a coarser granularity. At this level, each processor simulates a cell having multiple tracks on multiple integration levels. Once the capacity constraints at the edges of each level of each cell have been satisfied, the algorithm moves into a second phase, in which the exact routing within each cell is determined [18]. The first phase is best performed using a connectionist architecture, while the second phase requires more memory and more irregular calculation corresponding to detailed routing of each cell, and hence is best performed by using more traditional programming.

### *SMPs as cells in a cellular architecture*

In making his case for the Connection Machine, Hillis [15] argued that each processing element in the machine had to be small and simple. He claimed that it would be unreasonable to assume both that "there are plenty of processors" and that "there is plenty of memory per processor." That may indeed have been the case in 1985 when his thesis was published. However, gigascale technology has the potential to change this assumption in a dramatic way. Before 2011 it should be possible to build a very powerful cellular system of thousands of processors using a package about 20 cm on a side (**Figure 7**), each containing 1000 superscalar processors with 256 GB of DRAM and delivering a peak of 20 Tflops per package. Each of 64 chips in the package is a 16-way SMP like that shown in Figure 5.

Several advantages accrue from the use of an SMP as a node in such a cellular architecture:

- The node could be an off-the-shelf commodity item that had been built for a larger-volume commercial market. This not only reduces the cost of the system, but also takes advantage of the possibly higher development investments in that market.
- Program development tools including compilers and debuggers do not have to be rewritten from scratch. In addition, programs and libraries written for the SMP could run unchanged.
- An exciting consequence of the SMP as a node is that it is not constrained by the "one processor per item" concept inherent in the connectionist architecture. When the interconnection topology is known and regular, one could employ the sheer number of processing elements to solve the problem with the connectionist model. It is still possible to exploit the more standard Von Neumann model at levels that do not exhibit such regularity.

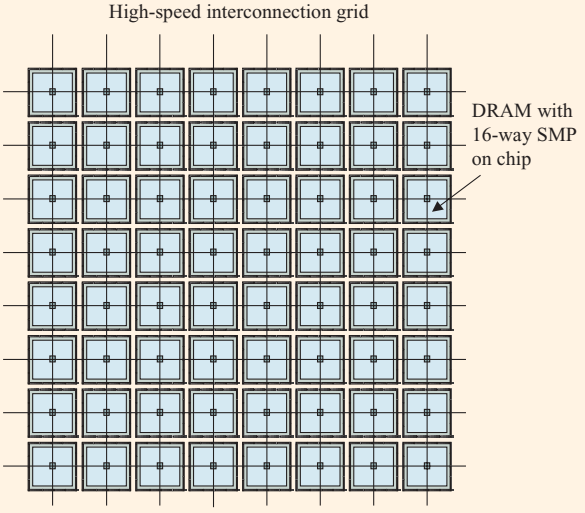


Figure 7

Cellular computer of 2011 containing chips shown in Figure 5. The chips are connected using a high-speed grid on a square package less than 20 cm on a side. Note that most of the package is DRAM—the configuration shown provides a peak of 20 Tflops along with 256 GB.

### *Improving system reliability and yield through redundancy*

Cellular computers also address another aspect of computing that will assume much greater importance in the future: reliable computation. Designers are perpetually trying to increase the functionality of chips by squeezing more transistors into the same area. At higher levels of integration, the reliability of the system measured in mean time between failures decreases. Unless redundancy measures are employed, a system employing billions of transistors cannot be expected to stay up for more than a few days at a stretch. On the other hand, because of the multiplicity of available processors, a cellular computer can continue to function with a reduced number of processors, provided system hardware and software are configured with this in mind.

As the number of components in a system increases, defects in manufacturing have a higher probability of causing the failure of a component and hence an entire chip. DRAM designers have long countered exactly such a problem by supplying redundant cells which were normally switched off, but which could be switched on to take over the function of cells damaged during manufacturing. The simple cell structure, along with the regularity of organization in a DRAM, made this an attractive and feasible solution and enormously reduced manufacturing costs. Cellular computers offer the same possibility. The

simplicity and regularity of a cellular computer make it easier to reconfigure a chip during manufacturing to work around areas with defects.

Improving chip yield or system reliability by exploiting redundancy hinges on the ability to test chips and to pinpoint the locations of faulty components. Chueng et al. [19] outline the challenges of testing large circuits in deep-submicron technology and point out inadequacies of current approaches. Greater reliance will be placed on the incorporation of built-in self-test (BIST) structures, which test components of the chip during system bring-up or during idle times between computations. It is also likely that new schemes will be developed which utilize the computing power of one or more processors to test another processor on a chip and to reconfigure around a faulty processor or section of a chip. Such autonomous testing and reconfiguration schemes are consistent with the cellular computing model outlined above, and will be critical in the development of large, highly available cellular server systems.

### *Reconfiguring around chip defects*

As fabrication costs rise, and as it becomes more difficult to manufacture perfect chips, architectures that tolerate defective components will gain in importance. The ability to easily reconfigure systems that have simple, regular structures could be taken to its logical extreme by making each computing element as primitive as possible, and by providing easy means of communication among these primitives. Scientists at the Hewlett-Packard Laboratories recently demonstrated a prototype of such a system [20]. The Teramac, as it was called, was built entirely from field-programmable gate array (FPGA) chips that were rejected during manufacturing because of defective components. Each chip was tested to identify defective components. A program then configured the remaining good components in all of the chips into a working computer.

The component-level redundancy employed by the Teramac may prove useful for future computation structures such as molecular computing or quantum computing, which are likely to contain significantly higher numbers of defective components. With silicon-based systems, as long as systems and application software are written the way they are today, redundancy at a higher granularity, e.g., at the processor level, will be easier to implement and more cost-effective than the Teramac approach of employing redundancy at the transistor-gate level. Useful processors need not be large; as indicated earlier in the paper, in technologies of the future each will be a very small fraction of the total chip area. Replication of such small processors on a chip could easily provide the needed redundancy to ensure yield and reliability.

## System-on-a-Chip

An SMP-on-a-chip offers one alternative to a billion-transistor uniprocessor. Another use for a billion transistors on a chip is to integrate functions that today are outside the processor chip, thereby ensuring reduction in size, cost, and power consumption of the system as a whole. In addition, reduced communication costs between the integrated elements could provide greater overall system performance compared to what could be obtained by improving processor performance alone.

The integration of diverse functions on a chip has been more common in high-volume consumer devices, where power consumption and packaging costs are important considerations. Today such integration comprises a variety of functions, including support for graphics, video, audio, wireless functions, modems, controllers for buses, hard disks, networking, and accelerators for network packet analysis. The integration of multiple types of components is dependent on the ability to mix diverse technologies on the same chip: high-performance logic, low-power logic, static RAM, rf, analog, and even DRAM. High-performance processors have not followed this route because the performance of traditional logic technology is degraded when components of another technology are integrated on the same die. Each technology requires some unique processing; hence, the cost of producing and testing these chips also increases with the incorporation of each technology, thus limiting the number of technology types on a chip. Despite these difficulties, the SIA roadmap notes that the number of designs using this style of integration, called System-on-a-Chip (SoC), will continue to grow. It will also be possible before the end of the decade to integrate flash memories and even more sophisticated technologies such as microelectromechanical systems (MEMS), ferroelectric RAM (FRAM), and a variety of chemical and optical sensors on the same die.

The versatility of SoC integration comes at a cost in designing, verifying, programming, and testing these chips. In order to make SoC more cost-effective, it is important to allow a designer to snap together items from a library of carefully designed and tested components. By designing these components in a consistent way, it becomes possible not only to increase the variety of circuits that can be constructed, but also to create easily configurable software development tools such as compilers, debuggers, and operating systems. **Figure 8** shows an embedded controller [21] designed around the IBM CoreConnect architecture using the Blue Logic* core technology [22]. The processor is a PowerPC 405 running at 266 MHz. The processor core contains about four million transistors occupying about 2 $mm^2$ in 250-nm technology[6] and dissipates about

[6] A new version of the core in 180-nm technology runs at 380 MHz and occupies an area of 1.4 $mm^2$.

0.5 W. The entire system sits on a die about 50 $mm^2$ in area and dissipates 1.1 W. This system illustrates the versatility of the modular building-block approach. While the performance of this processor is lower than that of a high-end general-purpose processor in the same technology, it boasts better performance per unit area and per watt dissipated, and vastly lower design cost. These advantages will only be accentuated with advances in lithography—such a design will occupy less than 1 $mm^2$ by 2011, leaving plenty of room on a really tiny chip for other parts of a system and for different types of transducers.

The ability to quickly integrate a customized, verifiable, and programmable system on a chip could make the SoC style of design attractive for a large range of applications. The success of this approach, however, will hinge on the availability of libraries of well-designed processor cores and other function-unit macros, and of tools to design, verify, and program the integrated system. In order to make the programming environment familiar to the programmer, processors in these systems are likely to use the same instruction-set architectures (ISAs) that will be prevalent on general-purpose processors. Eventually, though, the advantages offered by SoC in terms of cost, area, power, reliability, and modularity will likely drive greater reuse of many components from prevailing SoC libraries even in general-purpose processors.

While it seems attractive to put large sophisticated system functions all on the same die, most SoC applications will take advantage of available technology to reduce size and power consumption. Assuming continued progress in battery technology, minute but computationally powerful SoCs will find important uses in many areas, particularly in industrial and medical applications.

## Wire length and clock skew considerations

Reporting on their experiments with microarchitecture scaling for technologies up to 35 nm, Ho et al. [23] show that even under optimistic assumptions, the delay of a fixed-length wire on a chip will increase as feature sizes become smaller. If, on the other hand, the wire length itself is scaled with technology, its delay appears to become smaller, and approximately tracks the decreasing delay of transistor gates themselves. Ho et al. suggest that the complexity of designing future chips could be reduced by partitioning a chip into small domains, each of which has interconnections using short local wires internal to the domain, with long multi-cycle global wires used exclusively for communication between domains.

A similar conclusion about the need to partition logic on a chip is reached when one considers the clock skew problem in circuits. Most processors today employ a synchronous or clocked design approach with a centrally generated clock providing the synchronization in signals needed to ensure correct functionality of the logic. The wires in the clock distribution network cause clock skew—an edge of the clock does not appear at exactly the same time at all points on the chip. As mentioned earlier, even at fixed lengths, the delay, and hence the clock skew, becomes worse with smaller feature size. Thus, a smaller fraction of each clock cycle remains available for useful computation. A potential solution to this problem is to have multiple clock domains on a chip, each with a clock distribution network that is independent of those of the others. Communication between domains either may be synchronized at lower clock rates, or may be self-timed through some form of handshaking. Such handshaking protocols are not widely employed on a chip at present but are common at the next level of integration, namely between chips on a package or module. Shrinking lithography, which initially promises the availability of a large number of transistors on chip for greater chip functionality, eventually brings with it the same problems (and similar solutions to the problems) of communication that existed between chips on a module in some earlier generation.

Both architectures suggested in this paper embody the modular design concepts needed to overcome these wiring and clock distribution limitations. An SMP-on-a-chip inherently partitions the chip into areas that contain multiple processors and multiple memory banks. Each processor and each bank could form a natural domain both from local wiring and from clock distribution points of view. Partitioning is inherent also in the SoC concept.

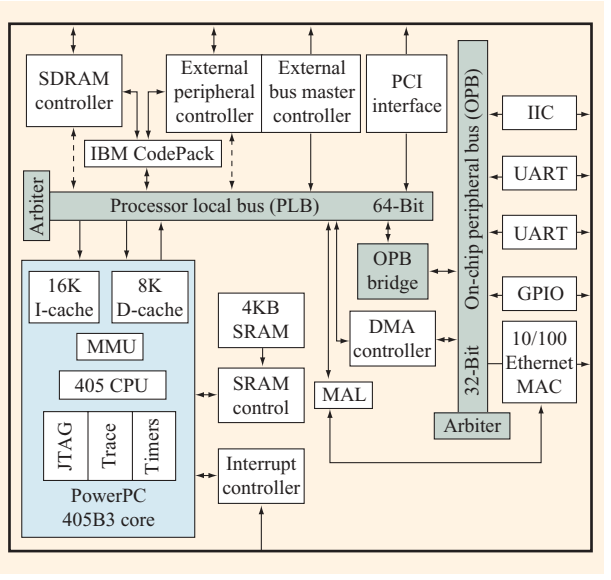


Figure 8

Block diagram of a contemporary embedded controller depicting wide use of core elements from a predesigned library.

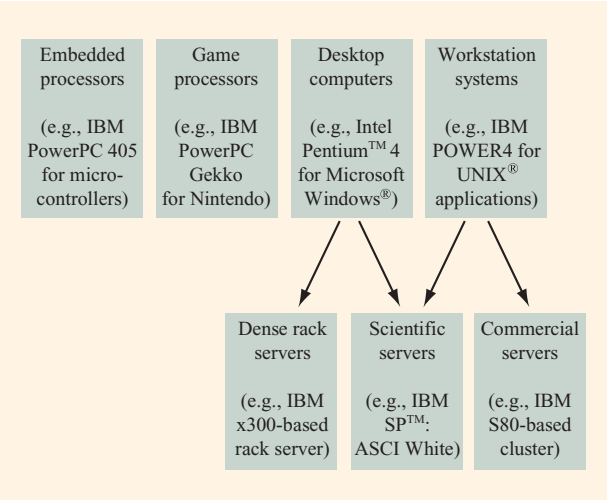


**Figure 9**

Typical processor reuse in different application areas. High-volume designs are shown on top row. Arrows indicate reuse of high-volume processor designs in high-end systems.

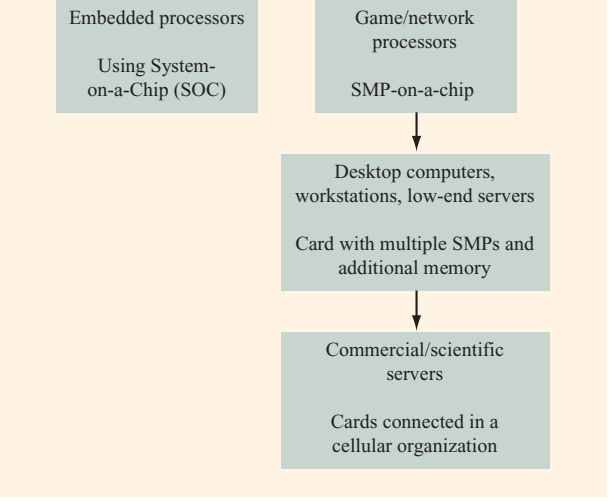


**Figure 10**

Projected processor reuse in a decade. High-volume designs are shown on top row. Second row shows reuse of high-volume game/network chips in cards designed for workstations and low-end servers. These cards could be reused in very-high-end servers shown in third row.

As SoC designs become larger and faster, however, it will be necessary to devise interfaces that elastically accommodate the possibly diverse clock requirements of the various components attached to them.

## Convergence of processors

In the past, it was common to design a new processor or even a new instruction-set architecture for a specific application area. Vector processors, RISC processors, mainframe processors, personal computing processors, signal processors, and graphics processors—each category coexisted with others, and each had a defined market and competing implementations. The boundaries between these areas have become less defined; processors today find use in areas other than the ones for which they were originally designed. As shown in **Figure 9**, processors for the large-volume home or office desktop market are identical to those used in engineering workstations. Commercial multiprocessing servers are often just systems employing larger numbers of such commodity processors. Large clusters use switch networks to interconnect hundreds of lower-cost cards incorporating these low-cost processors; clusters such as these are replacing special-purpose scientific systems of the past. Rack-mounted "dense" servers use a similar philosophy, except that they reduce the cost further by simply using LAN-type interconnections instead of expensive switches.

With a single-chip SMP of the type described in this paper, the industry could see further consolidation, as suggested in **Figure 10**. An SMP-on-a-chip could become a low-cost commodity item, for example by becoming the workhorse of the game industry. Low-cost desktop/workstation systems could be built using one or more such chips. Such a box would have sufficient power for many server applications, especially those needed in small-business environments. At the next level of the system hierarchy, a low-cost interconnection technology could emerge as the fabric for a cellular organization using these commodity SMP chips. Such a cellular system could serve as the common platform for both the high-end commercial and scientific markets, replacing today's disparate rack and cluster systems with their unique interconnection designs. This scenario would consolidate the market for all processors except the embedded microcontroller market, which is already moving to adopt the SoC paradigm.

## Concluding remarks

This paper has attempted to examine the effect that gigascale technology could have on the architecture of computer chips and computing systems. The day when a processor chip will have a billion transistors is not too far away; recent trends and industry projections lead one to believe that such a chip could exist before 2008.

There is an inherent inertia in software development models, and in computing paradigms in general, and this makes it difficult to argue that a fundamentally new way of computing will take hold less than a decade from now. On the other hand, scaling does appear to be less

promising at the microarchitecture and system levels than at the lithography level. Thus, it is unlikely that microarchitectural structures on a chip will simply be bigger and more powerful versions of structures that exist on chip today. The diminishing returns from increasing the width of today's largest superscalar chips or from increasing the number of processors in today's largest SMPs will lead to shifts in programming paradigms that are fundamental, though still based on knowledge gathered and tools developed over the last several decades.

Once technology allows placement of as many as 16 processors on a single chip, the focus of on-chip organization can shift from providing more computation power to bringing memory right alongside the processors. The amount of computation that could be delivered by a chip that has more than 4 GB of DRAM with 16 attached processors at 10 GHz would satisfy the requirements of most multimedia and game applications, and yet would be versatile enough to be a node in a commercial server cluster. Cluster computing is young, but is already beginning to make an impact in the server world. Its usefulness stems from the fact that it presents a scalable distributed computing view at a high level, but still allows a familiar shared-memory view at lower levels. The redundancy in such systems also makes them more tolerant to defects and failures, an important issue in large gigascale chips.

We have argued in this paper that another systems approach that offers a dual coupling paradigm is a cellular architecture that has connectionist properties at high levels but is based on a low-level cell that is an SMP-on-a-chip. Such a system can show good performance on algorithms cognizant of the relatively large delays in communicating between processing elements, yet its SMP nature at the low end provides a convenient model of programming for all existing applications as well as for those applications which cannot be readily structured in the connectionist style.

The System-on-a-Chip approach will make great strides in the coming years and could become the common way of exploiting gigascale technology. What makes SoC so exciting is the range of design options it offers. The ability to create an entire system by choosing items from a menu and by allowing software to integrate them on a chip promises options to appliance and instrument designers that can only be dreamt about today. The degree of integration afforded by gigascale technology reduces engineering and packaging costs for such applications. The ability to integrate a large variety of sensors and transducers directly beside the digital processor will change human life and society in remarkable ways.

There appear to be no insurmountable roadblocks to designing large, powerful computation structures on a chip in the gigascale technology. Yet, it is likely that ultimately the greatest benefit of gigascale technology may be in the blending of computers as inconspicuous components in common tools and appliances used by society, much as happened with a stalwart from another age, the electric motor. The computer industry has witnessed several disruptions in which a technology has crept up from low-cost platforms to become pervasive in high-performance platforms [24]. Gigascale technology may be the next in this long line of disruptive technologies. Rather than drive process technology as in the past, high-performance computer systems may eventually simply become beneficiaries of a technology driven by their miniaturized counterparts. In fact, the myriad uses that could be made of miniature processing engines lead one to confidently dismiss apprehensions about whether corporations and nations will invest the huge sums needed to develop this technology.

## Acknowledgment

The author wishes to thank Jim Smith, Monty Denneau, Eric Kronstadt, and Dan Prener for many useful discussions and for valuable feedback on earlier versions of this manuscript.



## References

1. C. J. Anderson, J. Petrovick, J. M. Keaty, J. Warnock, G. Nussbaum, J. M. Tendler, C. Carter, S. Chiu, J. Clabes, J. DiLullo, P. Dudley, P. Harvey, B. Krauter, J. LeBlanc, P.-F. Lu, B. McCredie, G. Plum, P. J. Restle, S. Runyon, M. Scheuermann, S. Schmidt, J. Wagoner, R. Weiss, S. Weitzel, and B. Zoric, "Physical Design of a Fourth-Generation POWER GHz Microprocessor," *IEEE International Solid-State Circuits Conference*, *Digest of Technical Papers*, February 2001, pp. 232–233.
2. *International Technology Roadmap for Semiconductors*, 1999 Edition, Semiconductor Industry Association, 4300 Stevens Suite Blvd., Suite 271, San Jose, CA 95129.
3. D. Burger and J. R. Goodman, "Billion-Transistor Architectures," Guest Editors' Introduction, *Computer* **30,** No. 9, 46–49 (September 1997).
4. Y. Patt, S. J. Patel, M. Evers, D. H. Friendly, and J. Stark, "One Billion Transistors, One Uniprocessor, One Chip," *Computer* **30,** No. 9, 51–57 (September 1997).
5. M. H. Lipasti and J. P. Shen, "Superspeculative Microarchitecture for Beyond AD 2000," *Computer* **30,** No. 9, 59–66 (September 1997).
6. J. E. Smith and S. Vajapayem, "Trace Processors: Moving to Fourth-Generation Microarchitectures," *Computer* **30,** No. 9, 68–74 (September 1997).
7. K. Olukotun, B. A. Nayfeh, L. Hammond, K. Wilson, and K. Chang, "The Case for a Single-Chip Multiprocessor," *Proceedings of the Seventh International Symposium on Architectural Support for Parallel Languages and Operating Systems*, October 1996, pp. 2–11.

8. G. Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," *Proc. AFIPS* **30,** 483–485 (1967).
9. R. Nair, "Dynamic Path-Based Branch Correlation," *Proceedings of the 28th Annual International Symposium on Microarchitecture*, *MICRO-28*, November 1995, pp. 15–23.
10. D. E. Lenoski and W.-D. Weber, *Scalable Shared-Memory Multiprocessing*, Morgan Kaufmann Publishers, San Francisco, CA, 1995.
11. E. Kronstadt, "Some Observations Based on Simple Models of MP Scaling," *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, Austin, TX, April 2000, pp. 123–128.
12. G. F. Pfister, *In Search of Clusters*, Prentice-Hall, Inc., Upper Saddle River, NJ, 1998.
13. J. M. Nick, B. B. Moore, J.-Y. Chung, and N. S. Bowen, "S/390 Cluster Technology: Parallel Sysplex," *IBM Syst. J.* **36,** No. 2, 172–202 (1997).
14. J. von Neumann, "First Draft of a Report on the EDVAC," University of Pennsylvania Report for the U.S. Army Ordnance Department, 1945.
15. W. D. Hillis, *The Connection Machine*, MIT Press, Cambridge, MA, 1985.
16. IBM Blue Gene team: F. Allen, G. Almasi, W. Andreoni, D. Beece, B. J. Berne, A. Bright, J. Brunheroto, C. Cascaval, J. Castanos, P. Coteus, P. Crumley, A. Curioni, M. Denneau, W. Donath, M. Eleftheriou, B. Fitch, B. Fleischer, C. J. Georgiou, R. Germain, M. Giampapa, D. Gresh, M. Gupta, R. Haring, H. Ho, P. Hochschild, S. Hummel, T. Jonas, D. Lieber, G. Martyna, K. Maturu, J. Moreira, D. Newns, M. Newton, R. Philhower, T. Picunko, J. Pitera, M. Pitman, R. Rand, A. Royyuru, V. Salapura, A. Sanomiya, R. Shah, Y. Sham, S. Singh, M. Snir, F. Suits, R. Swetz, W. C. Swope, N. Vishnumurthy, T. J. C. Ward, H. Warren, and R. Zhou, "Blue Gene: A Vision for Protein Science Using a Petaflop Supercomputer," *IBM Syst. J.* **40,** No. 2, 310–327 (2001).
17. S. J. Hong and R. Nair, "Wire Routing Machines—New Tools for VLSI Physical Design," *Proc. IEEE*, pp. 57–65 (January 1983).
18. R. Nair, S. J. Hong, S. Liles, and R. Villani, "Global Wiring on a Wire-Routing Machine," *Proceedings of the 19th Design Automation Conference*, Las Vegas, NV, June 1982.
19. K.-T. Chueng, S. Dey, M. Rogers, and K. Roy, "Test Challenges for Deep Sub-Micron Technologies," *Proceedings of the 37th Design Automation Conference*, Los Angeles, CA, June 2000, pp. 142–149.
20. R. Amerson, R. Carter, W. B. Culbertson, P. Kuekes, and G. Snider, "Teramac—Configurable Custom Computing," *Proceedings of the 1995 IEEE Symposium on FPGAs for Custom Computing Machines*, April 1995, pp. 32–38.
21. PowerPC 405GP Embedded Processor; Product Brief, *http://www-3.ibm.com/chips/products/powerpc/chips/*.
22. IBM Blue Logic Core Program: ASIC Core Descriptions, *http://www.chips.ibm.com/products/asics/products/cores/corelist.html*.
23. R. Ho, K. W. Mai, and M. A. Horowitz, "The Future of Wires," *Proc. IEEE*, pp. 490–504 (April 2001).
24. J. L. Bower and C. M. Christensen, "Disruptive Technologies: Catching the Wave," *Harvard Business Review*, pp. 43–53 (January–February 1995).

*Received June 6, 2001; accepted for publication February 20, 2002*

**Ravi Nair** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (nair@us.ibm.com).* Dr. Nair graduated from the University of Illinois with a Ph.D. degree in computer science in 1978; he has worked at the IBM Thomas J. Watson Research Center since then. He spent a sabbatical year at Princeton University in 1987–1988 and has also taught at Columbia University. Dr. Nair has worked in the areas of computer architecture, design automation, and testing, and has several publications, patents, and IBM awards in these areas. His current interests include processor microarchitecture, dynamic compilation, and virtual machine technology. Dr. Nair is a member of the IBM Academy of Technology and a Fellow of the IEEE.