Chips, Architectures and Algorithms: Reflections on the Exponential Growth of Digital Signal Processing Capability

Mark A. Richards[1]

School of Electrical & Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0250

Gary A. Shaw[2]

MIT Lincoln Laboratory
244 Wood Street
Lexington, Massachusetts 02420-9185

ABSTRACT

The celebrated 1965 prediction by Gordon Moore regarding exponential improvements in integrated circuit density is so widely known, and has proven so accurate, that it has been elevated to the status of a "law". Some pundits have even touted the unprecedented growth in IC performance as a major contributor to the sustained economic boom of the late twentieth century. Less well known and appreciated is the fact that many areas of computation have benefited equally from progress in algorithms. In this paper we compare and contrast the contributions to progress in signal processing capability of "hardware" (integrated circuits and computer architectures) and "software" (functionality and algorithms). While acknowledging the enormous impact that Moore's Law growth in hardware performance has had on signal processing applications, we make the case that algorithm improvements, in the form of improved functionality and implementation, have contributed significantly to past progress as well. We discuss important differences in the ways that progress in the hardware and software realms contribute to signal processing capability, and describe how signal processing application developers tend to draw successively on each different domain of improvement as their product designs evolve over time. Finally, we argue that sustaining exponential growth in signal processing applications in the future depends more than ever on sustaining innovation in software and algorithm development.

INTRODUCTION

While many of the important theoretical underpinnings and foundational algorithms of digital signal processing (DSP), for example the Nyquist sampling theorem and the fast Fourier transform (FFT), are rooted in the work of 19th century mathematicians and physicists, DSP emerged as a recognized field of study with the publication of the first DSP textbook by Gold and Rader in 1969 [1]. Coincidentally, it was in this same time period that Gordon Moore, then a pioneer of integrated circuit (IC) design at Fairchild Semiconductor, was asked during a keynote speech to comment on the likely course for this new technology. Moore's development team had completed a 32 transistor IC in

[1] E-mail: mark.richards@ece.gatech.edu

[2] E-mail: shaw@ll.mit.edu

1964 and a 64 transistor IC in 1965. Based on this scant historical data, Moore conjectured that it should be possible to double the number of transistors, and therefore, in some sense, the IC performance every year [2]. In a 1965 paper he argued that the number of "components" per IC could double every year through at least 1975 [3]. In 1975, with 10 years of experience in designing chips of exponentially increasing complexity, Moore amended his prediction at the annual International Electron Devices Meeting of the IEEE to a doubling every two years [4]. By the late 1970s the rate appeared to have stabilized at a doubling every 18 months. It is this variant that is perhaps most commonly recognized as Moore's Law [5],[6]. It has stood the test of time, with exponential improvement in IC technology the rule for more than 40 years. However, as will be seen shortly, a doubling period of two years appears to be a better long-term estimate of the growth rate of computing capability.

Even with the remarkable improvements in IC technology, many important applications for DSP would not be viable today without commensurate reductions in the computational complexity of foundational signal processing algorithms. Many more capabilities would not exist were it not for the emergence of entirely new signal analysis and processing paradigms. This paper examines the relative impact of improved IC technology and computing architectures, or more generally computing "hardware", versus fast algorithms and new mathematical techniques (computing "software") in advancing the capability of digital signal processing.[3] We describe the different ways in which progress in hardware and software tends to impact signal processing capability, and how application developers have historically drawn on these different types of progress at different points in product maturation. The 40-year history of DSP hardware and algorithm development highlights the value of continued investment in signal processing algorithms, and also provides a perspective for conjecture regarding the future evolution of DSP technology.

GETTING MORE FROM SIGNAL PROCESSING

Signal processing is the art and science of extracting actionable information from signals representing phenomena of interest. These may be real-time signals from any of a variety of sensors such as video, audio, seismic, electromagnetic and biomedical; or they may be constructed signals such as stock price series. Signal processing is performed using signal processors; today, this generally means a digital machine, usually programmable. The capability of a signal processor is determined by its "hardware" and "software". By "hardware" we mean the physical implementation, which includes both individual ICs and the system architecture. "Software" is the computational procedure, which includes both the mathematical functionality and the particular algorithm by which it is implemented. As an example of the distinction between functionality and algorithm implementation, the discrete Fourier transform (DFT) is a particular mathematical

[3] Some of our comments and examples regarding both hardware and software pertain to, or are drawn from, the realm of high performance technical computing, but apply equally well to signal processing.

function, a transformation of one data sequence into another with different and useful properties. The fast Fourier transform (FFT) is a particular sequence of computations that implements the DFT efficiently. Figure 1 illustrates this decomposition of a signal processor into IC devices (ICs) plus architecture, and functionality plus algorithms.

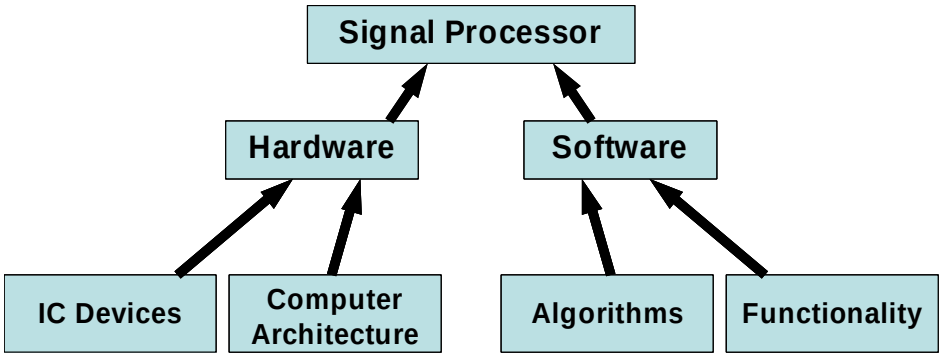

Figure 1. "Hardware" and "software" elements of a digital signal processor.

Continued research and technology development in signal processing are motivated by the ever-present demand to derive *more* information from signals. "More" can mean at least three different things:

- Faster: Derive more information per unit time; or
- Cheaper: Derive information at a reduced cost in processor size, weight, power consumption, or dollars; or
- Better: Derive higher quality information, for example, higher precision, finer resolution, higher signal-to-noise ratio, or reduced compression losses.

"Faster" can be achieved in two ways. A faster hardware approach may simply execute existing algorithms on a machine capable of more operations per second. This strategy is a direct beneficiary of denser, faster ICs and of architectural innovations such as multiple instruction execution techniques, memory hierarchies and, in larger-scale applications, multiprocessor architectures and communication fabrics. A software approach may adopt a new algorithm that implements a mathematical function in fewer operations (and with different precision, memory requirements, and quantization noise properties). Even the most basic DSP functions provide ample opportunity for diverse implementations.

For example, convolution with a finite-impulse response (FIR) filter impulse response can be implemented with a single straightforward convolution sum, frequency domain "fast convolution", or overlap-add or overlap-save methods (which in turn can be implemented in the time or frequency domain) [7]. The variation in arithmetic operation count for alternate implementations can easily be on the order of 5:1, depending on data and filter lengths. For infinite impulse response (IIR) filters, difference equations provide an efficient way to implement convolution with infinitely long kernels without

approximation. In fact, the focus of Gold and Rader's pioneering DSP text [1] is largely efficient digital equivalents to standard analog filtering using mostly IIR filters.

Improvements in signal processing hardware and software are not necessarily independent. While we usually evaluate algorithm efficiency in terms of mathematical operation counts, it is increasingly the case that, at least for more demanding computations, algorithms must be explicitly matched to the architecture of a multiprocessor system, and even to the internal architecture of microprocessors, in order to achieve the highest speeds. Conversely, multiprocessor architectures, especially communication fabric topologies, are sometimes designed to maximize the execution speed of particular algorithms they will host. For example, hypercube interconnection architectures are ideal for computing FFTs, while various block-cyclic data distributions are well-matched to many vector-matrix computations. Matching of architectures and algorithms does not reduce arithmetic operation counts, but it can significantly reduce the overhead requirements of memory access and data communication.

At first glance, more efficient ("cheaper") signal processing appears to depend primarily on improvements in hardware technology. Improved semiconductor technology allows the same operations to be implemented in less space, less weight, with lower power consumption, or at lower dollar cost. However, software also contributes to more efficient implementations. Conventional fast algorithms reduce operation counts and thus power consumption, but it is possible to go far beyond this to use a variety of coordinated software and hardware techniques to reduce power consumption [8].

As with "faster" and "cheaper", "better" may also be achieved by both software and hardware approaches. Improved devices may enable sampling with more bits of resolution and thus reduced quantization noise at a given sampling rate, or may compute an algorithm in floating-point instead of fixed-point arithmetic in real time. Software improvements come from new algorithms that obtain higher quality results from the same data as older, established procedures. A good example is the introduction of model-based spectrum estimation techniques such as autoregressive moving average (ARMA), the multiple signal classification (MUSIC) algorithm, or eigenspectrum techniques. Compared to classical Fourier methods, these techniques achieve higher spectral resolution from limited data; some allow enforcement of *a priori* constraints. Another example from very high resolution radar imaging is the development of range migration and tomographic (back projection) imaging algorithms, and more recently techniques based in part on modern spectral estimation. These techniques achieve finer resolution over larger image areas than older range-Doppler algorithms.

An important subset of "better", perhaps deserving of a separate category, is methods that extract new or different types of information from signals of interest. As technology evolves, entirely new domains of application are sometimes opened, or entirely new ways of solving existing problems become feasible. In this regard, "better" includes the discovery and application of entirely new functionality; for example, a novel function or transformation on the data that reveals information previously unobtainable by means of existing functions and methods. Examples of such "new math" for signal processing include wavelet techniques for nonstationary signal analysis, the Viterbi algorithm for

signal coding, and the emerging field of nonlinear signal processing. Whereas "faster" and "better" come from progress in hardware and in the algorithm aspect of software, entirely "different" information is obtained via innovations in the functionality aspect of software.

Historically, different technical communities have contributed to improvement of the different components of a signal processing application. Hardware progress is primarily the province of solid-state physicists, materials scientists, IC designers and computer architects. In the software arena, mathematicians, engineers, computer scientists, and algorithm developers take the lead. As we shall see, continued progress in signal processing may become increasingly dependent upon cooperation between these historically disparate communities.

In the remainder of this paper, we review some of the historical examples of Moore's Law at work and consider its impact on the performance growth of computing systems. We then provide an anecdotal review of the impact of algorithm developments over the same era, showing that both hardware and software progress have been critical to improving our signal processing capability over the last 40 years. We describe some important differences in the nature of progress in these two sides of the signal processing coin, including inherent differences between algorithmic versus hardware improvements. More important, however, is the unique capacity of new functionality to enable applications that could not be realized solely by improvements in the hardware or efficiency of existing algorithms. This leads to a short discussion of the development cycle by which progress in signal processing applications tends to occur. Finally, we speculate on the cost of maintaining historical rates of progress in hardware and software, concluding that continued, perhaps even increased, investment in algorithms is both essential and cost-effective if exponential growth in signal processing capability is to be sustained.

GROWTH IN SIGNAL PROCESSING PERFORMANCE

THE CONTRIBUTION OF HARDWARE

The capability of the physical machine implementation of a signal processor, which we refer to by the shorthand "hardware", is affected by the performance of the individual ICs that comprise the processor, memory, and communication elements, as well as the architecture that defines the overall organization of these elements. In this section we consider the rates of progress in ICs and computer systems.

Moore's Law

As Figure 2 illustrates, exponential improvement in the *cost* of computation was occurring even at the beginning of the twentieth century. Mechanical computers were supplanted by vacuum tubes and then transistor computers. In the early part of the twentieth century, performance doubled every three years [9]; but a change in the slope of the long-term growth in computing capability began in 1958-1959 with the invention

of the integrated circuit by Robert Noyce and Jack Kilby [10]. The IC rapidly became the basis of modern electronics.

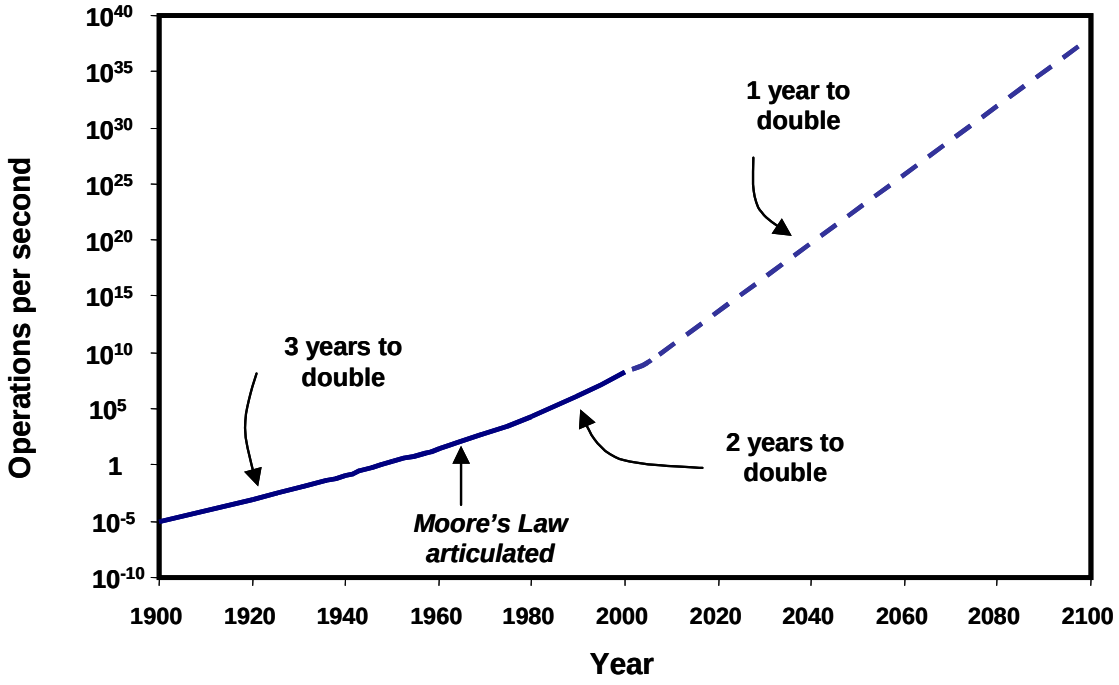

Figure 2. Retrospective and forecast of the computational capability available for a fixed price of $1000, (after Kurzweil [9]).

As originally stated, Moore's Law addressed the number of "components", primarily transistors, on an integrated circuit [3]; the density or performance per unit cost was not addressed directly. As understood today, Moore's Law predicts a doubling in circuit density, and concomitant improvement in performance, about every 1.5 to 2 years.[4] This is equivalent to 1.5 to 2 orders of magnitude every decade, a time scale more appropriate to the long view taken in this paper. Figure 3 shows that the density of Intel microprocessors, measured in transistors per chip, has closely followed this prediction for 30 years. The average growth rate of the data in Figure 3 is 1.5 orders of magnitude per decade. The related metric of clock rate has also followed an exponential growth curve, albeit a slower one; Figure 4 shows that since 1978, the clock rate of the same Intel microprocessors has increased at an average rate of one order of magnitude per decade.

[4] Tuomi presents a contrarian view that Moore's Law is too often extrapolated to aspects of computing beyond its originally intended scope of component density, and that many of these extensions are poorly supported by evidence, or ignore more important market forces on technology development.

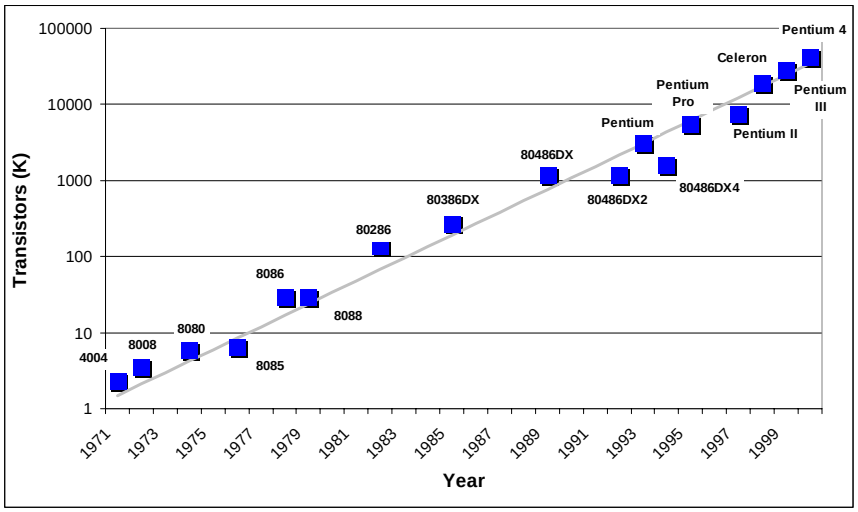

Figure 3. Growth in IC density as evidenced in Intel microprocessors. The solid line corresponds to an overall growth rate of 1.5 orders of magnitude per decade. Data from [11].

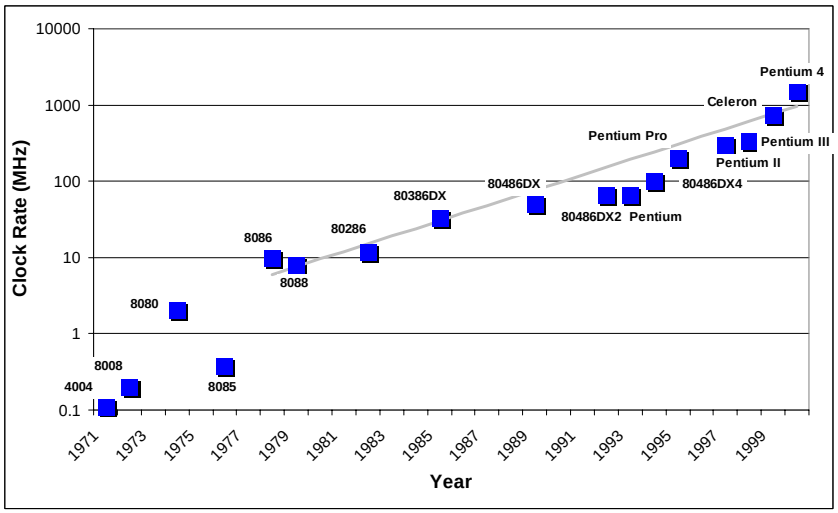

Figure 4. Increase in IC clock rate as evidenced in Intel microprocessors. The solid line corresponds to an overall growth rate of one order of magnitude per decade since 1978. Data from [11].

Signal processing practitioners are concerned with computational performance. We should therefore ask if either IC density or clock rate correlates directly with DSP computational performance. Because traditional core DSP algorithms are dominated by sum-of-products calculations, practitioners have long focused on the number of mathematical operations per second as a simple but meaningful indicator of processor performance. Since early digital processors typically were much slower at multiplication than at addition, the number of multiplications per second was often considered the important speed metric. With the 1982 introduction of the Texas Instruments TMS32010 DSP, it became common for DSP microprocessors to incorporate dedicated units to

perform multiplications in a single cycle, so that both additions and multiplications became of equal importance in evaluating processor speed for DSP. This in turn made IC clock rate a good indicator of arithmetic performance for a given algorithm. Thus, Moore's Law growth in IC density, and the concomitant increase in clock rates, has been a useful proxy for growth in DSP computational capability, at least over the last 20 years.

In recent years, the emergence of a variety of processor architectures featuring innovations such as very long instruction words (VLIW) and multiple instruction issues per cycle have muddied the significance of clock rates. In general, the exponential improvement in processor speed, with its attendant pressure on system communication and memory access, has now made memory hierarchy, parallel architecture, and communication fabrics equally important in estimating computing *system* performance, which we consider in the next section.

IC Minimum Feature Size Scaling

While Moore's Law has held sway for some 40 years, the continuing shrinkage of feature sizes that are at its root has a number of consequences that will eventually limit further progress in CMOS-based devices. Although development of finer-resolution mask making equipment remains a major challenge, thermal and economic considerations may arise as limiting factors sooner. For example, from 1970 to 2003, wafer exposure systems have risen in cost from about $20,000 to over $10,000,000, while wafer fabrication facility costs have risen from less than ten million dollars to over two billion dollars[5] [13]. Productivity has risen even faster, so that facility cost per unit output has dropped rapidly. Nonetheless, the absolute cost of a new wafer fabrication is reaching a scale that may be beyond the reach of even the largest semiconductor companies, requiring the formation of consortia or even of industry-government partnerships.

However, power and its attendant thermal considerations may pose the most imminent threat to Moore's Law. Transistor density scales as Δ^2, where Δ is the minimum feature size. To a first approximation, the underlying trends in power, power density, clock frequency, and energy per instruction can also be expressed in terms of minimum feature size. Table 1 summarizes the relationships, assuming constant die area and supply voltage. For example, the efficiency of microprocessors in terms of energy per instruction (nJ/instruction; equivalently, power per unit throughput in mW/MIPS) scales in proportion to Δ. This trend, sometimes referred to as "Gene's Law" after Gene Frantz of Texas Instruments [14], is illustrated in Figure 5, which shows that efficiency of a variety of microprocessors and DSP chips has improved at a rate of about 1.7 orders of magnitude per decade.

[5] Fab costs are yet another example of exponential increases, in this case at a rate of about 1.2 orders of magnitude per decade.

Table 1. Impact of Feature Size Δ on Microprocessor Metrics

Performance Metric	*Geometrical Dependency*
Clock Frequency	$1/\Delta$
Transistor Power	Δ
Transistor Density	$1/\Delta^2$
Total Device Power	$1/\Delta$
Power Density	$1/\Delta$
Energy per Instruction	Δ^2

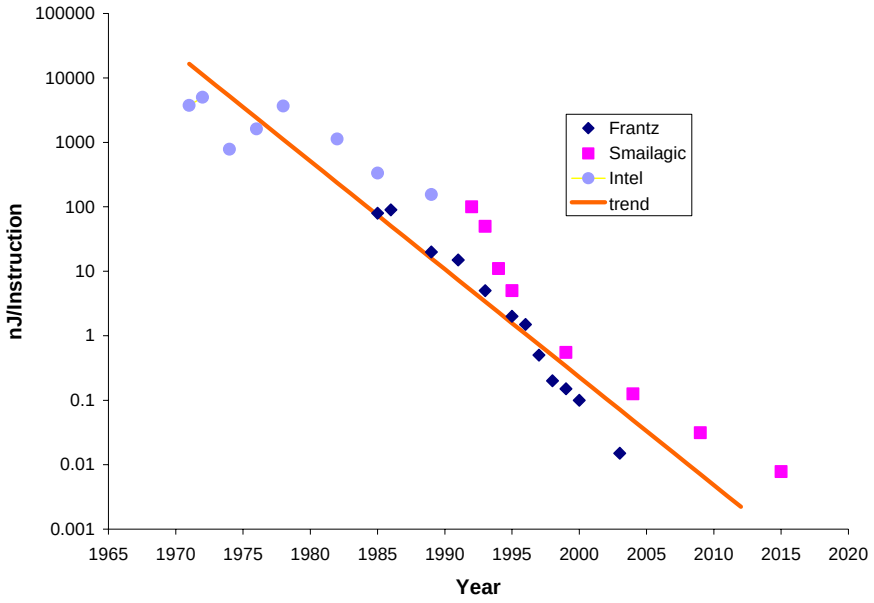

Figure 5. Energy per instruction for various processors. Data from [14],[15],[16].

Unfortunately, as indicated in Table 1, total power for a given die size *increases* in inverse proportion to feature size. Figure 6 shows the evolution of the power consumed by Intel microprocessors when running a high-power application [17]. The growth rate of this data is about 24% per year, or just under one order of magnitude per decade. Gunther *et al* show that the cost of cooling these devices rises rapidly when the power dissipation approaches 70 W per chip, primarily because heat sinks must be discarded in favor or more elaborate heat pipes or other technologies. Thus, there is a major cost incentive to keep power dissipation below 70 W per chip for desktop-class machines. The breakpoint would be much lower for small embedded processors.

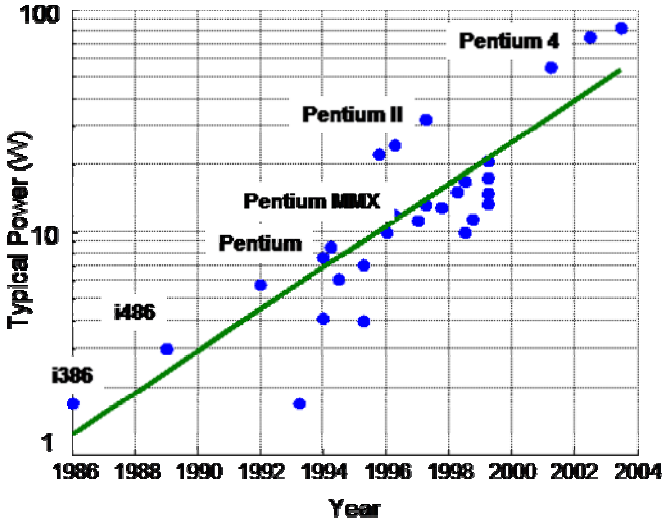

Figure 6. Historical evolution of power dissipation in Intel microprocessors. Data from [17].

Figure 7 shows the growth in power density, which exhibits the same trend evident in Figure 6. Overlaid on the trend curve of Figure 7 are several lines that provide power densities for a few non-electronic reference systems. Zhirnov *et al* argue in [18] that fundamental limits on the ability to remove heat as the power density increases will soon put an end to the simultaneous scaling of both clock speed and density, instead forcing them to be traded off: higher densities will require lower speeds to limit the power density; higher speeds will require lower densities. Since many current and emerging DSP applications are oriented toward wireless, mobile users, the development of low-power systems is receiving much attention from both algorithm researchers and device developers.

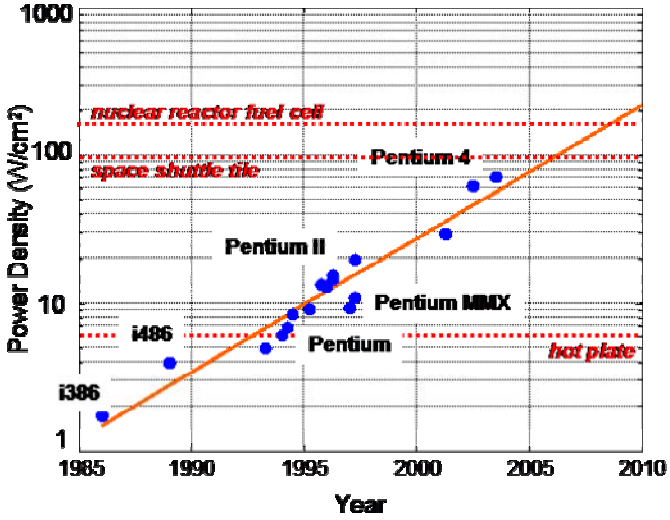

Figure 7. Power density of Intel microprocessor family Data from [17],[19].

Architecture

The exponential improvement in processor speed, with its attendant pressure on system communication and memory access, has now made memory hierarchy, parallel architecture, and communication fabrics equally important contributors to computing *system* performance. Historically, computing system performance has in fact improved exponentially, though not always at as high a rate as microprocessors. For example, in a 1993 book, Edward Yourdon claims a 20 to 30% improvement per year in "hardware technology", equivalent to 0.8 to 1.1 orders of magnitude per decade [20]. Patterson and Hennessy [21] describe a rate of computing system performance improvement of 1.1 orders of magnitude per decade in the 1970s, increasing to about 1.3 as the microprocessor revolution matured. In a 1984 column [22], Jon Bentley cites work showing that from 1945 to 1985, supercomputing hardware increased in speed by about factor of about 6×10^5, or just over a more optimistic 1.4 orders of magnitude per decade. All of these examples are below the Moore's Law rate of growth for ICs alone, suggesting that computing systems as a whole did not improve as quickly as their component ICs during this time.

Recently, progress in computationally efficient architectures at both the microprocessor and system levels has eliminated this lag. Continuing in [21], Patterson and Hennessy show the rate of improvement at the microprocessor level growing to 1.8 orders of magnitude per decade in the 1980s with the introduction of reduced instruction set (RISC) architectures. Numerous other sources and commentators also show that the growth in overall computing capacity now tracks, or even exceeds, Moore's Law. Sun Microsystems co-founder Bill Joy stated in 2001 that, starting in 1987, the rate of microprocessor performance improvement had increased from 35% per year in its first 15 years to about 55% per year [23], equivalent to a doubling every 18 months and 1.9 orders of magnitude per decade. This increase in the *rate* of exponential speedup, coincident with the switch to RISC microprocessors, is attributable more to the effects of architectural change in the microprocessor and memory hierarchies than to the semiconductor process improvements that drive Moore's Law.

Another measure of system performance, the Top 500 supercomputer list [24], confirms that this acceleration is also occurring at the computing system level. The rankings are based on speed in computing the LINPACK floating point linear algebra benchmarks. Figure 8 shows performance improvement rates for supercomputers over a decade ending in mid-2004 of 2.9 orders of magnitude per decade for the fastest computer at a given time (denoted by the curve labeled N=1), to 2.5 orders of magnitude per decade for the 100^{th}-fastest machine (N=100), to 2.8 (N=500) orders of magnitude per decade. This increase in the *rate* of exponential improvement of computing systems is due, at least in part, to progress in system-level architecture and not solely to microprocessor speedups.

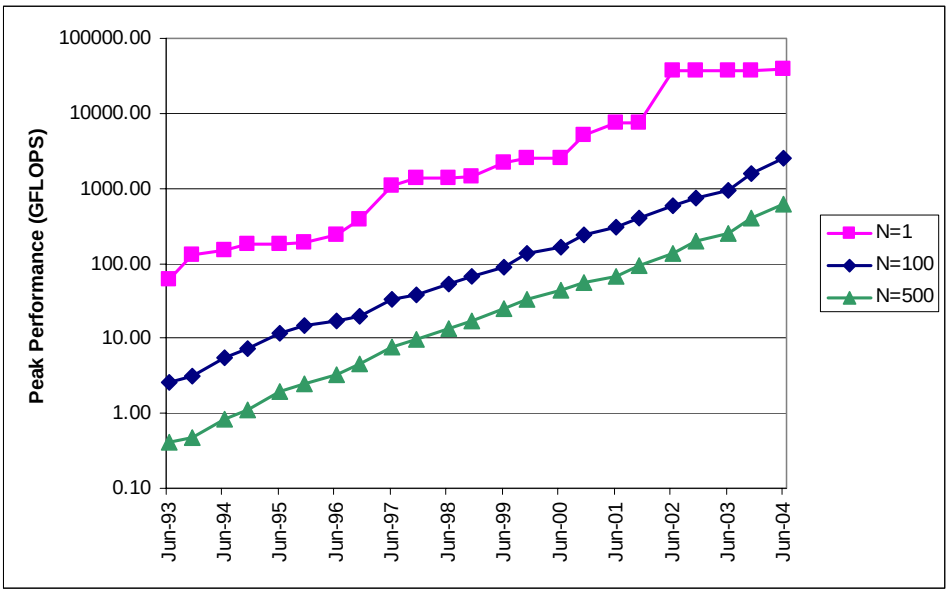

Figure 8. Performance growth of the 1st (N=1), 100th (N=100), and 500th (N=500) place computer in the Top500 supercomputer list. Data from [24].

THE CONTRIBUTIONS OF SOFTWARE

"Software," in the context of this paper, includes both the functionality to be implemented (the mathematics of the problem formulation), and the specific computational procedure to be followed (the algorithm). The choice of algorithms affects not only speed but also quality issues such as quantization noise and resolution, and can even affect other hardware metrics such as power consumption. Here, we focus primarily on speed. Given a specific hardware machine, and setting aside for the moment complications of multiprocessor architectures, we can expect that a signal processing algorithm's execution time is approximately proportional to its arithmetic operation count. Thus, operation counts are a simple proxy for algorithm speed.

Fast Algorithms

Have algorithm improvements kept pace with hardware improvements in contributing to the increases in computing and signal processing capability? To answer this question, we need to observe the performance of selected signal processing applications over a long period of time. Bentley [22] documents a scientific computing example, the solution of 3-D elliptic partial differential equations. He shows that during the four decades from 1945 to 1985, operation counts for problems computed on an NxNxN grid were reduced by a typical factor of $N^4/60$ through a succession of algorithmic improvements. For a broadly representative problem size of N = 64, the improvement is a factor of about $3\text{x}10^5$, just under 1.4 orders of magnitude per decade. Thus, the impact of algorithmic improvements on the time required for this class of scientific calculations over this period was similar to that of computing hardware improvements. Indeed, Rice ([25], p. 343)

states that "… the progress made through better methods from 1945 to 1978 exceeds the progress made through faster computers".

In a famous scientific computing example, IBM's "Deep Blue" supercomputer beat world champion Garry Kasparov in a chess competition in May 1997. Deep Blue was a special purpose machine with a peak processing speed of up to 40 trillion special-purpose operations per second and a design carefully matched to the algorithms to be used. It is difficult to compare Deep Blue to a general purpose machine, but Moravec [26] claims it to be equivalent to a general-purpose processor having throughput on the order of 1-3 trillion instructions per second (TIPS). Less than 6 years later, in February 2003, Kasparov played the then-champion "Deep Junior" computer chess program to a draw. The host computer, based on four conventional 1.9 GHz Pentium 4 processors, was capable of a peak throughput of approximately 15 billion instructions per second (GIPS).

If we consider the Deep Blue machine to be a 1.5 TIPS machine for arithmetic convenience, then the power of the machine needed to achieve the same functional result, namely playing chess on the level of Garry Kasparov, *de*creased by a factor of 100 in six years. Equivalently, the power of the algorithms used *in*creased by the same factor of 100 in six years. Hamilton and Garber [27] discuss some of the algorithm enhancements for generating and evaluating chess moves to unprecedented depths that led to Deep Blue's 1997 victory. Deep Junior, lacking the depth-of-search capability of Deep Blue's custom hardware, altered the evaluation strategy to encode more chess "understanding" and to incorporate a more aggressive strategy, including a willingness to sacrifice pieces [28].

A factor of 100 in six years corresponds to a rate of 2.15x per year, or 3.3 orders of magnitude per decade. As we have seen, the rate of increase in microprocessor clock rates is about one order of magnitude per decade. To achieve this 100x speedup from clock rate alone would have therefore required 20 years. Thus, the algorithm improvements accelerated the progress from the Deep Blue machine to the Deep Junior machine by 14 years. Even if we use the more broadly representative hardware improvement rate of 1.5 orders of magnitude per decade, the algorithmic improvements are responsible for a 7.3-year acceleration of progress.

There are many other examples of exponential growth in capability. The downstream speeds of dial-up modems of the sort used with personal computers provides a more modest example of exponential growth in signal processing capability for a particular function sustained over a significant period of time. These improvements were primarily algorithm-driven, taking advantage of advances in modulation and coding schemes. For example, binary phase-shift keying (BPSK) gave way to quadrature PSK (QPSK) or quadrature amplitude modulation (QAM), resulting in rates increasing from 300 bps to 1200 bps. The introduction of trellis coding and a 16-point constellation increased rates to 9600 bps; a 128-point constellation increased rates further to 14.4 kbps. Treichler, Larimer, and Johnson review the history of modem algorithm enhancements [29], showing that data rates over unconditioned lines increased from 2400 bits per second (bps) to 56000 bps in approximately 13 years, equivalent to an average of 23% per year, or just over 1 order of magnitude per decade. Eldering, Sylla, and Eisenach [30] provide

a similar review based more on the release dates of international modem standards, as illustrated in Figure 9. They show a more dramatic increase from 300 bps in 1986 to 56000 bps in 1998, a span of 12 years and a rate of 1.9 orders of magnitude per decade. If we start with the 2400 bps modem, their data gives a growth rate of 1.2 orders of magnitude per decade, somewhat more consistent with the Treichler *et al* data.

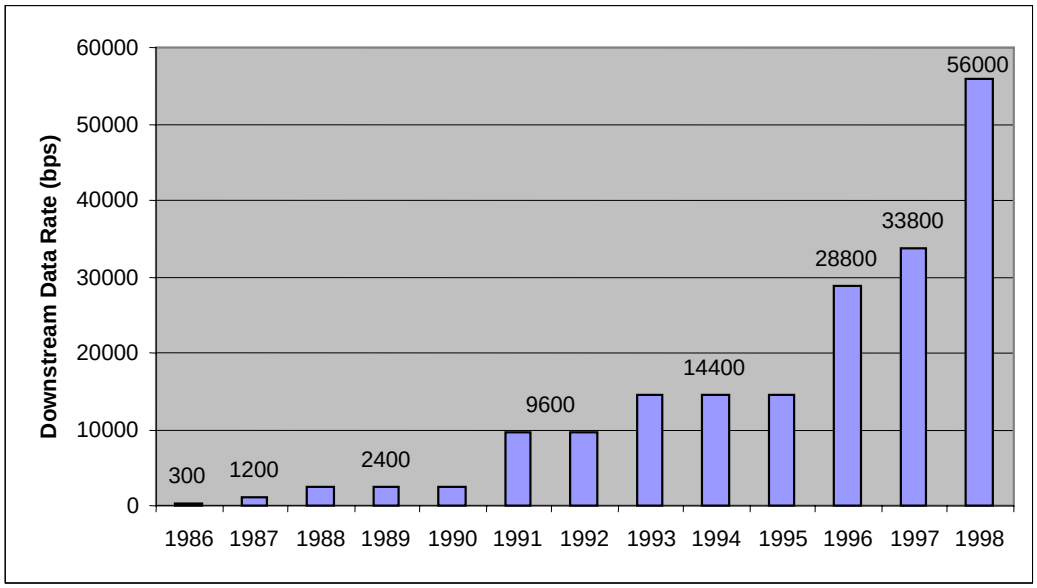

Figure 9. Growth in two-wire dial-up modem downstream data rates. (Data from Eldering, Sylla, and Eisenach [30])

Complete signal processing applications are generally composed of a number of component algorithms. The speedups discussed above are a composite of speedups of varying degree in the components of the overall application. It is therefore useful to look in more detail at improvements in specific component algorithms.

If there is a single canonical signal processing algorithm benchmark, it is the 1K (N = 1,024) complex FFT. Direct implementation of the DFT sum requires N^2 complex multiplications, or just over 10^6 for N = 1024. The original radix-2 Cooley-Tukey algorithm [31], published the same year (1965) as Moore's Law, reduces this to $(N/2)\log_2 N$, or 5120 for N = 1024.[6] This reduction by a factor of about 200 is equivalent to 15 years of hardware improvement at 1.5 orders of magnitude per decade! That is, publication of the FFT made it possible to compute a 1024-point DFT in an amount of time that would not have been achievable for another 15 years if one relied only on Moore's Law speedups of computer hardware. A radix-4 Cooley-Tukey algorithm

[6] More precise operation counts that include additions and account for trivial multiplies are available in [32].

further improves this incrementally, reducing the complex multiply count to $(3N/8)\log_2 N = 3840$, a reduction over the direct DFT by a factor of 273; this is equivalent to a little over 16 years of Moore's Law exponential hardware improvement. As shown in Figure 10, for longer FFTs even more years of hardware improvement are needed to effect the same reduction in computation time that the FFT affords over a direct sum-of-products DFT.

The FFT algorithm is not unique in achieving an exponential reduction in computational complexity for a core function. In 2000, the journal *Computing in Science and Engineering* published a special issue addressing "The Top 10 Algorithms" of the 20th century [33]. Two more outstanding examples of fundamental improvements in computation complexity drawn from this list are the Quicksort algorithm and the fast multipole algorithm. Quicksort reduces the average complexity of the problem of sorting N items into numerical order from $O(N^2)$ to $O(N\log N)$, the same $O(N/\log N)$ gain provided by the FFT over the DFT.[7] The fast multipole algorithm is even more powerful, reducing the complexity of N-body simulations from $O(N^2)$ to $O(N)$.

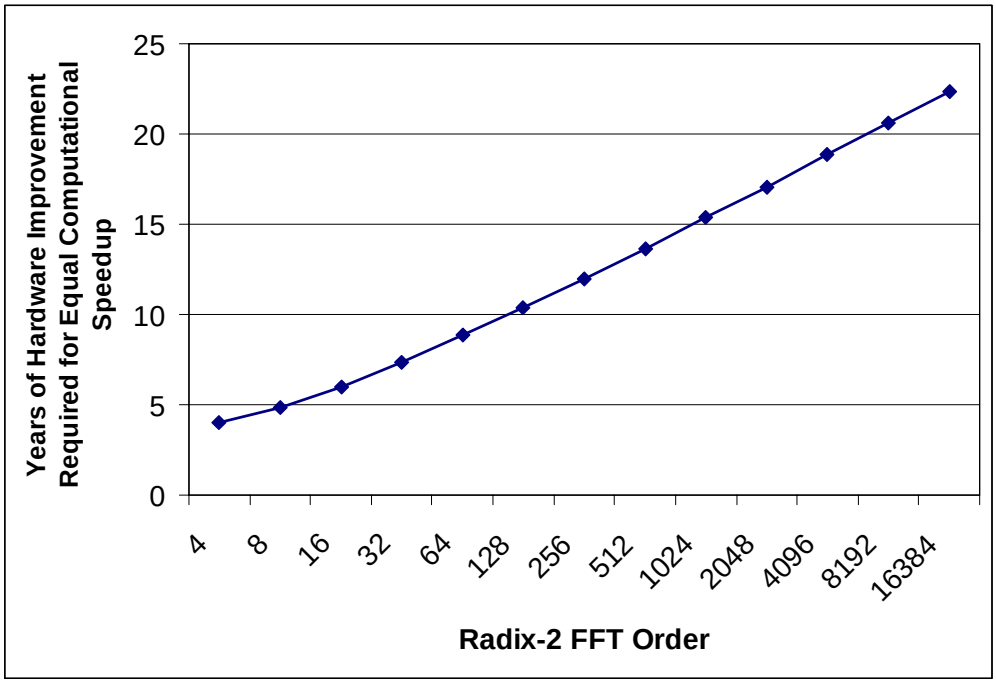

Figure 10. Years of Moore's Law improvement required to equal FFT computational savings relative to a sum-of-products DFT. Assumes hardware speedup of 1.5 orders of magnitude per decade (doubling every two years).

[7] The notation $O(N^2)$ means "order of N^2", *i.e.* proportional to N^2.

Another example of order reduction by a full factor of N is the solution of the autocorrelation normal equations used in parametric modeling. The classical algorithm for solution of the (N+1) equations resulting from an order N model is Gaussian elimination, which requires approximately $N^3/3$ multiplications. The first developers of linear prediction used the Cholesky method to solve the equations with half as many computations. Levinson's recursion, published in 1947, cut the number of multiplications to N^2+2N [34]. This is a major reduction in order from $O(N^3)$ to $O(N^2)$; in fact, the reduction of computational load by a factor of N is greater than that obtained for DFTs by the FFT algorithm.

The speedups in solving the normal equations, while important, have not had as dramatic an impact as has the discovery of the FFT. The reason is that the typical applications of linear prediction in signal processing involve relatively small order problems, limiting the gain in absolute terms. For example, linear prediction of speech typically involves an order of N = 8 to 12. For N = 10, the reduction from Gaussian elimination to the Levinson algorithm is a factor of 2.8; for N = 15, it rises to 4.4. At 1.5 orders of magnitude per decade, these improvements are comparable to 3 and 4.4 years of hardware improvement, respectively. One can wonder if there is a discovery yet to be made that will reduce the computation from $O(N^2)$ to $O(N\log N)$. Even if not, the Shur algorithm requires a similar number of operations but is more amenable to parallel implementation, opening up another path to improved performance [35].

Low Power Algorithms

Reduced computational complexity allows us to solve a given problem more quickly, or to solve a larger-order problem in the same amount of time. Such speedups are not always the goal of algorithm improvements. The exploding use of small, individual, but very powerful computing and communication devices such as portable audio devices, laptop computers, cellular telephones and personal digital assistants (PDAs) has drastically increased the importance of power efficiency. Power efficiency is being addressed at every level, from new sources such as micro fuel cells to low power circuit design in semiconductors, and algorithms can contribute here as well [8]. Classical algorithm speedups have a direct payoff in energy efficiency because they reduce the number of operations required to complete a computation. Thus, when an algorithm that reduces computational load is introduced, a slower or less dense processor can be used, reducing power consumption. Parallelization of algorithms is another approach to architecture and algorithm-based power reduction. Parallelization enables a computation to be done with multiple, slower processing units instead of one very fast one. The use of slower units allows reductions in chip supply voltage and, since power consumption is proportional to the square of voltage, the parallel approach will often consume less total power. Approximation is an algorithmic approach that trades off accuracy for efficiency. An obvious way to do this is by using fixed- instead of floating-point arithmetic. More sophisticated methods exist, for example the use incremental refinement algorithms that allow computations to be stopped early, or of lossy compression algorithms to reduce the amount of data that must be transmitted, processed, and stored.

HARDWARE *VS.* ALGORITHM IMPROVEMENTS

Considered in total, the evidence cited in the preceding sections suggest that a rate of increase in the performance of ICs and computer systems of 1.5 orders of magnitude per decade can be taken as representative. Furthermore, faster generations of computer hardware have been produced frequently and predictably for 40 years now, and may maintain this rate of improvement for at least another decade. If one considers the normalized system performance metric of operations per second at a fixed price point, then improvements accrue not only from shrinking IC feature sizes, but also from more mature fabrication, integration, and marketing processes. Kurzweil's prediction of price-point trends in Figure 2, which is based on this normalized performance metric, suggests many decades of improvement may still lie ahead.

In contrast, speedups in algorithms are usually manifested as one fundamental breakthrough such as the basic Cooley-Tukey algorithm, reducing the complexity of the computation as a function of the problem order (*e.g.*, from N^2 to $N\log N$), followed by a consolidation phase featuring a number of lesser improvements by factors of two or more. The example of the FFT shows that major improvements in algorithms can produce, in effect, instantaneous progress equivalent to a decade or more of improvement in signal processing hardware performance, while even lesser algorithm innovations (factors of 2) equate to a three-year advancement.

While equating algorithm improvements to an equivalent number of years of hardware improvements provides a common basis for comparison, it is important to recognize a fundamental difference between algorithm and hardware improvement. To first order, improvements in hardware speed translate directly to reductions in computational time, regardless of the algorithm. The benefits of faster hardware are immediately available to almost all algorithms, and the algorithm speedup is manifested directly in terms of reduced time for execution. In contrast, a particular algorithm innovation benefits only those applications that use that class of algorithms, and the speedup is usually a function of the dimension or order of the problem. While Bentley's data suggest that algorithm advances have occurred frequently enough to keep pace with hardware progress over the last 40 years, the longer Moore's Law persists, the more difficult it becomes for algorithm innovations to maintain parity with hardware speedups for a fixed problem order. Consider again the DFT example. While the development of a 1024 point radix-2 FFT pushed processing throughput ahead by a one-time leap of 15 years, if the dimensionality of the problem (in this example, the FFT size) does not change, continuing savings from algorithm improvements will be limited to more modest gains from such techniques as higher radices or algorithms matched to particular architectures. In contrast, faster microprocessors benefit both the DFT and FFT algorithms in equal proportion, and continue to do so at a predictable rate so that over time, hardware speedup will account for an increasing percentage of the cumulative reductions in computational time.

Furthermore, the impact of reductions in algorithm complexity is blunted when other processing that is not amenable to fast algorithms is considered. The impact of a fast algorithm is limited to the portion of an application that uses that algorithm. For

instance, if a 1K DFT computation accounts for 80% of the total runtime of an application, then the 200x improvement in the DFT occasioned by using the FFT will reduce the overall runtime by a factor of 4.9x, while if the DFT was only 20% of the original computation, the improvement is only a factor of 1.25x. In comparison, a speedup in individual processors, especially if balanced with improved memory and communication, benefits the entire application.

One way to summarize the complementary nature of speedups in hardware versus algorithms is to note that hardware speed increases exponentially and *predictably* as a function of time. As long as Moore's Law remains in effect, we can count on the fact that computationally complex algorithms not currently realizable in real time will eventually become realizable, and we can even predict approximately when! We only need wait a sufficient amount of time to acquire faster ICs that will support a real-time implementation. In contrast, while reduced complexity algorithms are discovered unpredictably in time, they increase execution speed exponentially and predictably *as a function of problem dimension or order*. Therefore, discovery of a fast algorithm acts in effect like the discovery of a "worm-hole" in time evolution of an application, with the "time-compression" benefit of the worm-hole increasing in proportion to problem order. The benefit is always instantaneous and sometimes startling, allowing real-time implementation of high-order problems long before they would be enabled by hardware improvements. Thus, a given algorithm improvement, while not predictable in the short term, and not sustainable in an evolutionary fashion over time, may nonetheless be enormously valuable.

These observations also suggest the reason that Moore's Law has become so widely known, even reaching the consciousness of the general public, while the contributions of algorithms to computing and signal processing capability seem to be less universally known or appreciated. The increase in hardware speed benefits all computing, from the most obscure scientific applications to the most common household PCs, and all problems from the very large to the very small, in roughly equal measure. The invention of a fast algorithm benefits only those users whose applications rely heavily on that algorithm, and even then, provides the most benefit to the highest order problems.

NEW FUNCTIONALITY

None of these considerations of computing speed address the most fundamental payoff of algorithm research: the development of entirely new capabilities resulting from the application of new concepts and mathematics, the "functionality" component of "software". Improvements in hardware speed enable application performance improvements by supporting more of the *existing* functionality within a given amount of time, space, power, or other resource. For example, a processor speedup might allow higher order algorithms (*e.g.*, bigger FFTs) to be applied, or a measured data set to be compared to a larger library of stored templates, within a given time constraint. Hardware speedups do not directly point to new functionality. In contrast, breakthroughs in functionality add entirely new tools to the signal processing toolbox.

Consider the example of speech recognition, a capability now available in shrink-wrap software; versions of it are used every day in such mass public applications as airline and banking telephone-based information systems. It is unlikely that speech recognition would have ever reached the desktop, no matter how powerful the computer, had it relied on the vocoder or pattern recognition techniques of the 1960s. Rather, its success was dependent upon adopting new approaches based on parametric modeling and hidden Markov models. The migration of world-class computer chess from special-purpose supercomputers to everyday desktop machines was made possible by a fundamental change in algorithmic strategy, abandoning brute-force exhaustive generation of all possible sequences of moves in favor of a less comprehensive but more sophisticated evaluation of potential moves. Continued research in both functionality *and* algorithms is critical because of their potential to produce sudden, large gains in computational capability and also to enable fundamental new capabilities not achievable through increased speed alone.

The impetus for entirely new algorithms comes about in at least two ways. The first, similar to the speech recognition application discussed above, is the response of researchers to the realization that the performance shortfalls of an existing algorithm suite are not due to speed limitations, but are fundamental functionality shortcomings. The second is the development of new application demands that open up entire new fronts of research. Development of the Internet and cellular networks vastly increased the demand for progress in data compression and other aspects of telecommunications, emphasizing implementations that are efficient in the use of computation, memory, and power.

MATURATION CYCLES IN APPLICATION IMPLEMENTATION

The implementation of a signal processing capability typically evolves and matures in a manner that draws at different times on the different mechanisms for achieving "more": ICs and architectures, functionality and algorithms. Early development of a new application idea is typically focused on proof of concept (better or different) rather than faster or cheaper. The developers are primarily concerned with finding a set of algorithms that work reliably to perform the desired task adequately. At this stage, the concern is with identifying and defining functionality, the "math" of the problem that will make the new application possible. Standard available software tools are used on commodity laboratory computing platforms, often without much concern (yet) for real-time performance. Thus, the initial implementations of a new capability rely on new functionality to achieve progress in signal processing.

Once it is shown that a new capability is possible and useful, the developmental emphasis often turns to improving the implementation, where improvements might include increased speed; reduced size, weight, or power; or reduced cost. This drive for better implementations brings both hardware and algorithms into play. Historically, Moore's Law has made it possible to predict when real-time implementations of a non-real-time systems could be realized, or when one could shrink the size or cost of the system by re-

implementing the now-known functionality on successive generations of computing hardware.

Any improvements achieved in algorithm efficiency are of the same fundamental value as equivalent improvements in hardware speed; but whether such improvements are available depends on the functionality of the application. For example, it seems unlikely that further substantial reductions in the operation counts of algorithms for computing DFTs are likely. Thus, if the new application relies heavily on the DFT, it is not a good candidate for major algorithmic efficiency improvements over current practice. While careful attention to algorithmic details may offer significant savings of factors of 2 and 3 over the original prototypes, order-of-magnitude improvements will come only from improvements in hardware performance. On the other hand, if the new functionality takes advantage of mathematical approaches for which good fast algorithms are not yet known, the application can potentially benefit from both hardware performance improvements and from algorithm efficiency breakthroughs. Even if this is not the case, if the functionality is compatible with cache-savvy and parallel implementations, opportunities exist for obtaining speedups through parallel algorithms that, while not reducing fundamental operation counts, do match the problem architecture to the processor architecture in ways that significantly improve efficiency. It is well known that a pipeline architecture is especially well-suited to relatively coarse-grain parallel implementation of an FFT. However, matching of the FFT algorithm to the hardware can be carried further, as for example in the FFTW library [36], which optimizes smaller components of the FFT algorithm to the cache structures of particular microprocessors used in each pipeline stage.

However, there will always come a day when all of the efficiencies that can be wrung out of the mathematics of an application have been realized. When that day comes, only hardware speedups will provide further improvements in the implementation of the application. If these are too slow to come, then the application must be improved by a fundamental shift to a new way of achieving the same end, *i.e.* to new math. For instance, no further improvements in conventional dial-up modem speeds have occurred since 1999; instead, progress has been achieved by a shift to a new telecommunication loop architecture, digital subscriber line (DSL). Such a paradigm shift then enables a new cycle of algorithm improvements to accompany the steady march of semiconductor improvements. This process of application development and evolution is illustrated in Figure 11. A new application concept is demonstrated in an initial proof-of-concept realization using a particular algorithmic approach and hardware design. Multiple cycles of hardware and algorithm improvements result in a series of faster, better, and cheaper implementations. When algorithmic improvements to the basic mathematical functionality are exhausted, Moore's Law continues to afford improvements in performance. However, at some point the drive for "more" will be stopped by fundamental limitations of the functional approach. At this point, a change to a different approach based on new math and physics is required, enabling new cycles of hardware and software improvements.

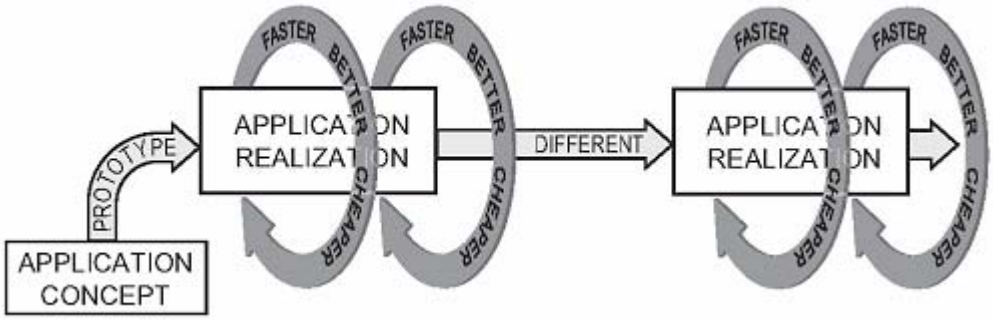

Figure 11. Notional illustration of the initial development and evolution of a signal processing application.

As another example, early vocoders, as they would be implemented today, are essentially filterbank technologies and as such could take full advantage of improvements in DFT algorithms and semiconductor technology. However, building ever faster and more compact vocoders would never achieve the improvements in speech coding and generation occasioned by the shift to predictive modeling, a fundamentally different representation of the signal. The predictive approach shifted the emphasis away from DFTs and onto matrix algorithms for the solution of the normal equations. This in turn brought opportunities for new algorithmic efficiencies through the Cholesky, Levinson, and Schur algorithms discussed earlier.

While much of the preceding discussion has emphasized reduced computational complexity and increased speed, progress in signal processing capability is not always in the direction of faster execution times. In many cases, the extra capability provided by faster hardware or algorithms is used not to make a given functionality run more quickly, but to allow the designer to implement a more complicated function in the same amount of time. That is, so long as real-time deadlines are met, functionality expands to occupy the processing power available. An example is the introduction of digital I/Q (in-phase and quadrature) filtering in coherent communication and radar receivers. This technology replaced analog mixing and filtering with digital techniques requiring a great deal of high speed digital filtering.[8] In exchange, the receiver designer achieves reduced I/Q channel mismatches, a major error source. (It is also worth noting that many clever algorithmic tricks have been developed to moderate the digital filtering load.) This form of progress is simply an example of choosing "better" over "faster".

INVESTMENTS IN SIGNAL PROCESSING PROGRESS

Computing in general and signal processing in particular have both benefited from, and been driven by, the forty-year reign of Moore's Law. In this paper, we have argued that

[8] Perhaps this is the quintessential example of Prof. Tom Barnwell's observation that DSP is "That discipline which has allowed us to replace a circuit previously composed of a capacitor and a resistor with two anti-aliasing filters, an A-to-D and a D-to-A converter, and a general purpose computer (or array processor) so long as the signal we are interested in does not vary too quickly."

signal processing applications have also benefited significantly from progress in algorithms, and that many applications could not have succeeded without the benefit of new mathematical concepts and reduced-complexity algorithms, no matter how fast the hardware. We maintain that the contributions of hardware and algorithms over time have been, in a broad sense, comparable. Yet we perceive that, relative to historical funding profiles, support for fundamental research in signal processing algorithms and mathematics is losing ground relative to investment in hardware technology. During the past decade in which we have seen investments in computer hardware rise to in excess of $220B annually, we have also witnessed the disintegration of some of the most distinguished commercial signal processing R&D organizations, as well as a shift in both civilian and military R&D from fundamental research to near-term products and demonstration systems, respectively. Because mathematical functionality and algorithm innovations have each provided a major share of the total progress in signal processing capability, it is reasonable to ask how much support should go toward research in DSP algorithms and functionality so that we can maintain our total rate of progress. While it is beyond both the scope of this article and the capability of the authors to accurately estimate the investments it took to get to our present state of hardware and software capability, we can develop a rough estimate of the costs to continue progressing at the same rates.

The capital cost of semiconductor fabrication plants has risen from about $6 million in 1970 to $2 billion for next generation facilities currently coming on line [5],[13],[37]. The construction of these facilities is an unavoidable cost of striving to stay on the Moore's Law curve for semiconductor improvements. In a more comprehensive estimate, the Semiconductor Industry Association (SIA) reports that combined capital and R&D investments in the semiconductor industry in the year 2001 alone totaled $31.3 billion [39]; and this is for the U.S. industry only. Is a similar investment required to maintain the historically observed rate of improvement in algorithms?

Before addressing this question, we note that while continued exponential improvement in DSP chip performance is wholly dependent upon smaller-featured devices fabricated in the new semiconductor fabrication facilities, the return on the enormous investment in these facilities is determined not by the DSP market, but by the much larger markets for memory chips, general purpose microprocessors, microcontrollers, and all the many other semiconductor devices that pervade the modern world. Because of this wider impact of semiconductor investments, we would be foolish indeed to suggest that government and industry should be making investments in DSP algorithm development equal to those made in device development and fabrication.[9] However, we conjecture that the investment to sustain algorithm innovation has been, and will remain, much less.

While a careful assessment of the cost of maintaining progress in algorithm innovations requires too great a depth of economic analysis to present here, consideration of direct salary costs of the signal processing community provides anecdotal support to our

[9] If, however, they should choose to do so, please contact the authors for instructions on where to send the check.

conjecture. The median salary in 2003 for a full-time academic senior researcher in signal processing algorithm and software techniques is approximately $100,000, or $120,000 for a similar researcher in the defense industry [38]. Including overhead costs, the total cost to support our researcher might be up to $300,000 per year. If the researcher is a professor, he or she might well be assisted by several graduate students; if employed in an industrial firm, by one or two junior researchers. We can postulate that for $600,000 per year we might, on average, be able to support a group of 4 researchers, making the average cost per researcher about $150,000 per year.

It only takes one person with inspired insight to develop a new technique that can improve performance by an order of magnitude or provide an entirely new approach to a problem. Since we can't know who that person is in advance, we can seek to improve our odds for a breakthrough by investing in many research groups. At the end of 2001, there were 18,487 members of the IEEE Signal Processing Society [41]. This may be only a small fraction of practicing signal processing engineers, but it is also true that only a small portion of the practicing signal processing engineers are likely to be involved in research leading to new algorithms. To focus on DSP engineers active in research, consider the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). The premier digital signal processing conference focused on new research, ICASSP attracts as many as 2,000 attendees, though surely many more researchers and students would like to attend. Considering these two figures, we can conjecture that the active DSP research community, at least in the U.S., might number on the order of 10,000 people. Using this as an estimate of the aggregate level of effort currently devoted to algorithm R&D, the funding needed to support and even to advance the rate of progress is in the neighborhood of $1.5 billion per year. Therefore, annual R&D funding for the signal processing research community in the U.S. amounts to only about 5% of the investment in semiconductor R&D and capital investment, and less than 1% of the total computer hardware investment. Considering what it has yielded over the years, the algorithm development side of signal processing progress is a bargain!

Furthermore, it is likely to become more of a bargain in the future. Whereas the capital costs of new wafer fabrication facilities double every 3 years, Figure 12 shows that median engineering salaries rise only 11% in the same amount of time. Thus, the cost of investing in algorithm research increases at a much slower rate than that observed for semiconductor fabrication facilities. Consequently, as time goes on, algorithm investments become more cost effective relative to capital investments in hardware technology.

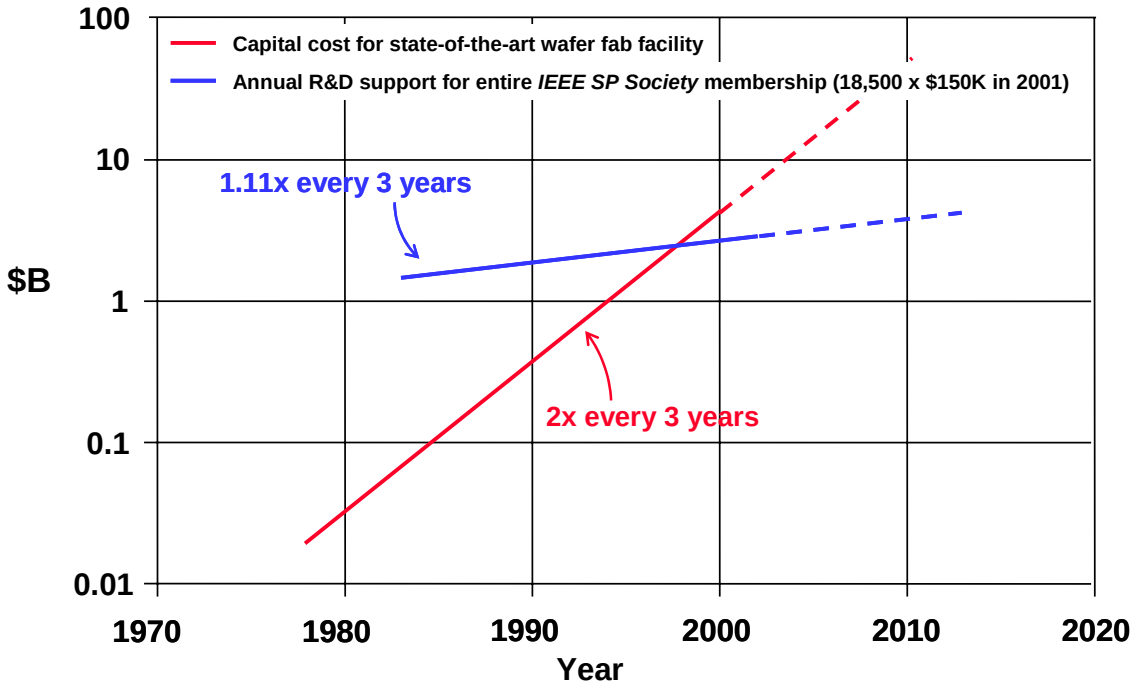

Figure 12. Relative rate of growth of investments in semiconductor fabrication facilities, and in signal processing labor costs.

FUTURE EVOLUTIONARY PATHS FOR DSP

How will progress in signal processing capability be sustained in the future? The opportunities for improvement are in IC devices and computer architectures, the "hardware" of the signal processor; and in algorithms and functionality, the "software". Historically, IC design and fabrication on one hand, and mathematics and new functionality on the other, have been largely independent endeavors. In many cases, processor architecture and algorithms are also developed independently. However, as researchers and developers strive for greater efficiency, architectures and algorithms are becoming inextricably intertwined, defining a new middle ground that makes hardware and software increasingly appear as a continuum as illustrated in the revised version of Figure 1, shown in Figure 13. Just as over time additions and multiplications came to have equal importance in evaluating DSP microprocessor performance, now cache hierarchies, memory access time, and data communication latencies and rates are becoming equally significant determinants of system speed. The increasing impact of these architectural features and thermal considerations upon the efficient use of modern processors is driving computer design increasingly to parallel multiprocessor systems, in some cases involving hundreds or even thousands of processors. The expectation that the performance increase of individual chips will level off due to power density considerations implies that growth in computing capacity must come instead from combining large numbers of possibly slower and less dense processors in highly parallel fabrics. This then puts a premium on the development of algorithms that are highly

parallelizable and closely tied to, or even adaptive to, complex multiprocessor architectures. In early efforts in this direction, several groups have developed algorithm libraries for specific functions that automatically optimize the details of architecture-dependent traits such as data block size (to match processor cache size) so as to self-optimize performance on a specific architecture. Examples include the FFTW library for FFT computations [36], the ATLAS system of linear algebra routines [42], and the SPIRAL system for DSP algorithms [43]. However, much more work is still needed to develop efficient algorithms for systems based on hundreds or thousands of processors.

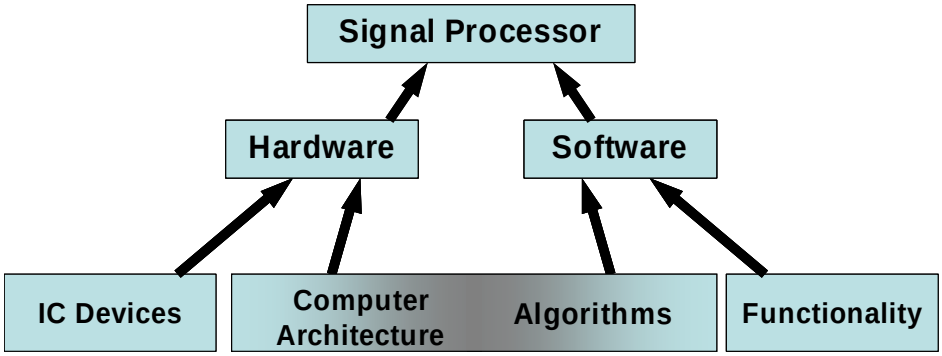

Figure 13. Evolution of Figure 1 to reflect the growing interdependence of algorithms and processor architectures.

FFTW and ATLAS attempt to optimize parameterized algorithms for implementation on existing processors. A complementary approach being pursued by several university and commercial design teams is examining new processor architectures that better match the characteristics of many signal processing applications. A representative example is Stanford University's "Smart Memories" project [44], which uses a tiled microprocessor architecture to overcome wire delay and communication latency problems in large chips while taking advantage of extensive parallelism and data locality often found in signal processing and multimedia applications. In essence, these architectures accelerate the incorporation of large-scale multiprocessor design techniques, such as multiprocessor clusters in various topologies with local memory and high bandwidth interconnection, multi-level memory hierarchies, and dataflow programming models to the single IC level.

In the search for fundamentally better functionality, algorithms may evolve in a number of new directions. Certainly multiscale algorithms such as the wavelet transform and the multipole methods mentioned earlier will be increasingly important. Novel conceptual frameworks such as "quantum signal processing" [45] extend classical linear algebraic computations. The best architectures for such procedures are still a matter of research. More fundamentally, nonlinear methods are finding increasing use, especially in image-oriented and some statistical applications. Traditional linear algorithms are based in many cases on sums of products as the fundamental computation. Nonlinear techniques instead make heavy use of non-arithmetic operations such as sorting and counting. This change could have profound implications for the design of successful signal processing architectures. At a minimum, nonlinear techniques move the computational emphasis

from floating point to fixed point operations, which are inherently faster and more energy-efficient.

Even farther reaching is the increasing interest in the development of "knowledge-based" signal processing, where traditional arithmetic algorithms are modified through the incorporation of external data sources that constrain or guide the problem solution, for example through data selection and editing. For instance, estimates of heterogeneous clutter interference statistics used in adaptive detection and tracking algorithms can be improved by editing and augmenting the radar data with information from geographic information systems (GIS) and other sources.

If the predictions in Figure 2 regarding the cost of computation hold true, new paradigms for developing and implementing signal processing algorithms will certainly be required. Implementing a sequential, von Neumann program in 10^{18} bytes of memory in the year 2030 hardly seems plausible, much less optimal. Whole new methods of developing, refining, and implementing signal processing algorithms will be required. A current example of where such new programming paradigms are needed is research in "cognitive technologies" for computing systems for applications such as mobile robotics and human-computer interaction. These new concepts in signal processing not only create new capabilities, they also create both the opportunity and the need for continued progress in algorithm innovations if we are to continue, or perhaps even increase, the phenomenal growth in signal processing performance.

SUMMARY AND CONCLUSIONS

Progress in signal processing capability is the product of progress in IC devices, architectures, algorithms and mathematics. It is well known and expected that hardware capability improves at a rate of 1.5 orders of magnitude per decade. Less appreciated is that, over the long term, the continuing cycle of new functionality and efficient algorithms has contributed a similar rate of improvement. Thus, the total progress in signal processing capability owes an equal debt to both the software side and the hardware side. To maintain the same rate of progress in the future, it is essential that we continue our investments in both areas.

There are sound reasons to maintain, or even to increase, our annual investment in DSP software R&D. The remaining lifetime of Moore's Law is a favorite topic of speculation, but many commentators believe that for silicon CMOS, it can be expected to hold for perhaps another ten years [6]. Others argue that the rate of growth will slow soon due to such issues as power consumption and wire delays on chips that are very large compared to the feature size [44]. Gordon Moore himself has recently predicted that a slowdown is imminent, though he did not attempt to quantify by how much [46]. When the exponential improvement in computing hardware does begin to falter, progress in new functionality and fast algorithms will provide the principal paths to increased capability.

The scope of research opportunities in algorithms is actually expanding. Many of the traditional examples of algorithm breakthroughs we have cited were new computational procedures that substantially reduced operation counts for a specific function; the

solutions of PDEs, sorting, and the FFT are all examples. New mathematical techniques provide new opportunities for similar improvements. Algorithms that achieve speedups by clever matching of mathematical problem structure to computer architecture represent a very different avenue of attack. More fundamentally, emerging research into knowledge-based and cognitive systems opens up to scrutiny entirely new types of both functionality and computational complexity.

There are physical limits that will eventually halt the exponential shrinkage of ICs. However, economic considerations may slow IC progress before the physical limits are reached [5],[6]. Thus, there is a real prospect of a slowdown in the half of signal processing progress contributed by improvements in semiconductor and microprocessor technology. On the other hand, we have given at least anecdotal evidence that the cost of maintaining and even growing an active and robust signal processing research community is a fraction of the investment needed to keep semiconductors on the Moore's Law growth curve. Relatively modest, less rapidly growing investments, are likely to maintain the contributions of algorithm and functionality research to the total progress in signal processing performance. Once the exponential growth that has been characteristic of Moore's Law begins to slow, we conjecture that significant increases in the level of investment in algorithm research will be needed to sustain the performance improvements we have come to rely upon. Thus, increasing investments in algorithm research may be a means of sustaining exponential growth in performance without the exponentially increasing costs associated with commensurate improvements in semiconductor technology.

ACKNOWLEDGEMENTS

In the course of developing this paper, the authors received encouragement and helpful comments from a number of their colleagues. They would especially like to thank Al Oppenheim, Charles Rader, Dennis Healy, Ron Schafer, James Anderson, Dan Campbell, Gene Frantz, Robert Graybill, and Sherman Karp.

This work was sponsored in part by the Department of the Air Force under Contract F19628-00-C-0002. Several of the ideas were developed in connection with programs sponsored by the Defense Advanced Research Projects Agency. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

REFERENCES

[1] B. Gold and C. M. Rader, *Digital Processing of Signals.* McGraw-Hill, New York, 1969.

[2] C. C. Mann, "The End of Moore's Law?," *Technology Review*, June 2000.

[3] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, v. 38(8), April 19, 1965.

[4] D. J. Yang, "The Lawgiver", sidebar in "Leaving Moore's Law in the Dust", *U. S. News & World Report*, vol. 129, no. 2, p. 38.

[5] G. D. Hutcheson and J. D. Hutcheson, "Technology and Economics in the Semiconductor Industry," *Scientific American*, Jan. 1996, pp. 54-62.

[6] R. R. Schaller, "Moore's Law: past, present, and future," IEEE *Spectrum*, pp. 52-59, June 1997.

[7] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd ed. Prentice-Hall, New Jersey, 1999.

[8] K. J. Ray Liu *et al*, "Algorithm-Based High-Performance Multimedia Signal Processing," *Proceedings* IEEE, vol. 86(6), pp. 1155-1202, June 1998.

[9] R. Kurzweil, *The Age of Spiritual Machines: When Computers Exceed Human Intelligence*. Viking Press, New York, 1998.

[10] J. S. Kilby. Modular Electrical Unit. U.S. Patent 3,052,822, Application filed May 28 1958.

[11] "Microprocessor Trends" at IC Knowledge web site, www.icknowledge.com/trends/uproc.html.

[12] I. Tuomi, "The Lives and Detahs of Moore's Law," *First Monday* internet journal, vol. 7(100), Nov. 4, 2002. Available at http://www.firstmonday.org/issues/issue7_11/tuomi/index.html.

[13] S. W. Jones, "Exponential Trends in the Integrated Circuit Industry", at IC Knowledge web site, www.icknowledge.com/trends/exponential.pdf.

[14] Gene Frantz, "Digital Signal Processing Trends," IEEE *Micro*, Nov-Dec 200, pp. 52-59.

[15] Smailagic, A., and Siewiorek, D., "System Level Design as Applied to CMU Wearable Computers", *Journal of VLSI Signal Processing Systems*, Kluwer Academic Publishers, Vol. 21, No. 3, 1999.

[16] Intel "Microprocessor Quick Reference Guide", www.intel.com/pressroom/kits/quickref.htm.

[17] S. H. Gunther *et al*, "Managing the Impact of Increasing Microprocessor Power Consumption," Intel Technology Journal, First Quarter 2001. Available at www.intel.com.

[18] V. V. Zhirnov, *et al*, "Limits to Binary Logic Switch Scaling – A Gedanken Model," *Proceedings* IEEE, vol. 91, no. 11, pp. 1934-1939, Nov. 2003.

[19] "System and Components Reference Guide," www.pcguide.com/cpu.

[20] E. Yourdon, *Decline and Fall of the American Programmer*, Prentice-Hall, New York, 1993, p. 268

[21] D. A. Patterson and J. L. Hennessy, *Computer Architecture: A Quantitative Approach*, 2nd ed. Morgan Kaufmann, San Francisco, 1990.

[22] Jon Bentley, "Programming Pearls," *Comm. ACM*, v. 27(11), pp. 1087-1092, Nov. 1984.

[23] W. N. Joy, "Reduced Instruction Set Computers (RISC): Academic/industrial Interplay Drives Computer Performance Forward," Computing Research Association position paper, www.cs.washington.edu/homes/lazowska/cra/risc.html.

[24] "Top 500 Supercomputer Sites" web site, www.top500.org. Data shown is through the 23rd biannual list, June 2004.

[25] J. Rice, *Numerical Methods, Software, and Analysis: IMSL® Reference Edition.* McGraw-Hill, New York, 1983.

[26] H. Moravec, "When will computer hardware match the human brain?", *Journal of Evolution and Technology*, vol. 1, 1998. Available at www.transhumanist.com/volum1/moravec.htm.

[27] S. Hamilton and L. Garber, "Deep Blue's Hardware-Software Synergy", IEEE *Computer*, pp. 29-35, Oct. 1997.

[28] P. Ross, "The Meaning of Computers and Chess", IEEE *Spectrum* "Web-Only News", March 1, 2003, www.spectrum.ieee.org/WEBONLY/wonews/mar03/chesscom.html.

[29] J. R. Treichler, M. G. Larimer, and C. R. Johnson, Jr., "Voiceband Modems: A Signal Processing Success Story", in "Highlights of Signal Processing for Communications" (G. B. Giannakis, editor), IEEE *Signal Processing Magazine*, vol. 16(2), March 1999.

[30] C. Eldering, M. L. Sylla, and J. A. Eisenach, "Is There a Moore's Law for Bandwidth," IEEE *Communication Magazine*, pp. 117-121, Oct. 1999.

[31] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Computation of Complex Fourier Series," *Mathematics of Computation*, vol. 19, pp. 297-301, April 1965.

[32] C. S. Burruss and T. W. Parks, *DFT/FFT and Convolution Algorithms*. Wiley, New York, 1985.

[33] J. Dongarra and F. Sullivan, editors, "The Top 10 Algorithms", special issue of *Computing in Science & Engineering*, January/February 2000.

[34] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, pp. 417-418, Prentice-Hall, New Jersey, 1978.

[35] D. S. Watkins, *Fundamentals of Matrix Computations*. Wiley, New York, 1991.

[36] M. Frigo and S. G. Johnson, "FFTW: An Adaptive Software Architecture for the FFT," *Proc. IEEE Intl. Conf. Acoustics, Speech, and Sig. Proc.*, vol. 3, pp. 1381-1384, 1998. Also see www.fftw.org.

[37] "Economic Trends" at IC Knowledge web site, www.icknowledge.com/economics/fab_costs.html.

[38] IEEE-USA 2003 Salary Survey, available through salary.ieeeusa.org.

[39] Semiconductor Industry Association web site, www.semichips.org/home.cfm.

[40] Survey of median engineering salaries, 1983-2003, U.S. Bureau of Labor statistics, www.bls.gov.

[41] "Membership Development Progress Report," IEEE Technical Activities Board, January 2002.

[42] R. C. Whaley, A. Pettitet, and J. J. Dongarra, "Automated Empirical Optimization of Software and the ATLAS Project", available at math-atlas.sourceforge.net/

[43] Markus Püschel, *et al*, "SPIRAL: A Generator for Platform-Adapted Libraries of Signal Processing Algorithms," *Intl. Journal of High Performance Computing Applications*, vol 18(1), pp. 21-46, Feb. 2004.

[44] K. Mai *et al*, "Smart Memories: a modular reconfigurable architecture," *Proceedings* 27th Intl. Symp. on Computer Architecture, pp. 161-171, 10-14 June 2000.

[45] Y. C. Eldar and A. V. Oppenheim, "Quantum Signal Processing", IEEE *Signal Processing Magazine*, vol. 19, no. 6, pp. 12-32, Nov. 2002.

[46] M. Kanellos, "More life in Moore's Law, creator says". CNET News.com, July 9, 2002, http://news.com.com/2100-1001-942671.html. The remark was made in an interview following the awarding of the Presidential Medal of Freedom to Dr. Moore.