Title: Sustaining the Exponential Growth of Embedded Digital Signal Processing Capability[1]

Authors:

Dr. Gary A. Shaw (*first author/corresponding author/presenting author*)
MIT Lincoln Laboratory
244 Wood Street
Lexington, Massachusetts 02420-9185
Voice: 781-981-7994
Fax: 781-981-4608
E-mail: shaw@ll.mit.edu

Dr. Mark A. Richards
Georgia Institute of Technology
School of Electrical and Computer Engineering
Atlanta, GA 30332-0250
Voice: 404-894-2714
Fax: 404-894-0560
E-mail: mark.richards@ece.gatech.edu

Candidate Sessions (drawn from HPEC abstract submission web site):

- Algorithm Mapping to High Performance Architectures
- Future Program Office Needs for Embedded Computing Technologies
- Case Study Examples of High Performance Embedded Computing

Author's Comments:

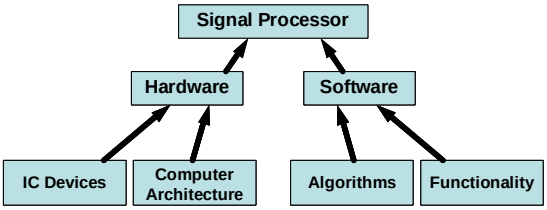
This proposed paper is less of a technical exposition than a retrospective and exhortation regarding the importance of sustained efforts in algorithm and software research and development. The paper compares and contrasts the contributions of hardware and algorithm improvements to the rate of improvement in high performance computing and signal processing. We conjecture that as IC shrinkage and attendant performance improvements begin to slow, the exponential rate of improvement we have become accustomed to for embedded applications will be sustainable only through a faster pace of improvement in algorithms and software.

[1] *This work was sponsored by the Department of the Air Force under Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.*

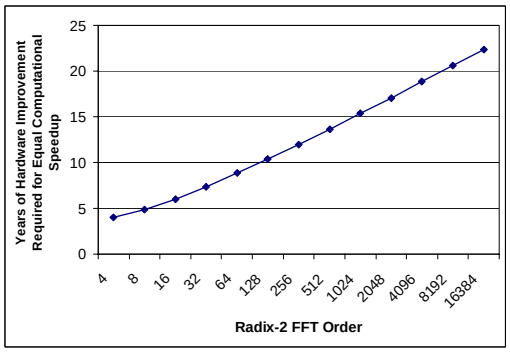
Sustaining the Exponential Growth of Embedded Digital Signal Processing Capability

Gary A. Shaw and Mark A. Richards

The exponential improvement in IC device density and throughput observed over the past 40+ years, first articulated by Gordon Moore in 1965 and canonized as Moore's Law, is well known and has had an immense impact on our society. Obvious, but less appreciated, is the fact that the computational capability of a digital signal processor, and more generally any high performance computing system, is the sum of its hardware capability and its software capability. 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. For example, the discrete Fourier transform (DFT) is a particular mathematical function, while the fast Fourier transform (FFT) is a particular sequence of computations that implements the DFT efficiently.



Observed over long periods of time, software improvements, particularly fast algorithms, have contributed exponential rate improvements that equal or exceed the improvements accruing from faster hardware. The best known example in the DSP community is, of course, the FFT. Depending upon the problem order, the computational speedup afforded by the basic radix-2 FFT is equivalent to anywhere from one to 20+ years of device improvement. The FFT is not unique in terms of its impact on the acceleration of HPEC applications. In an example of long-term evolution of scientific computing algorithms, Bentley[2] documents improvements in the solution of 3-D elliptic partial differential equations. He shows that from 1945 to 1985, operation counts for problems computed on an NxNxN grid were reduced through a succession of algorithmic improvements by a typical factor of $N^4/60$. For a broadly representative problem size of $N = 64$, the improvement is a factor of about $3\text{x}10^5$, or just under 1.4 orders of magnitude per decade. This rate of improvement is on par with that observed for IC devices. Other examples of long-term, as well as sudden, algorithmic improvements can be found in applications ranging from dial-up modems to computer chess. The improvements associated with these application areas will be illustrated in detail in the presentation.



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

While evidence exists for equal rate improvements from algorithms and hardware, there are nonetheless significant differences in the manner in which hardware and software improvements are manifested. Hardware throughput performance increases exponentially and *predictably* as a function of time. As long as Moore's Law remains viable, we can count on the fact that a computationally complex algorithm not currently realizable in real time will eventually become realizable, and we can even predict approximately when it will be realizable! 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*. Discovery of a fast algorithm thus acts in effect like the discovery of a "worm-hole" in time evolution of an application. Thus, rather than waiting for the time evolution of Moore's Law hardware improvements, the discovery of a fast algorithm creates an instantaneous leap ahead in performance, with the "time-compression" benefit of the worm-hole increasing in proportion to problem order.

In comparison to the historically predictable payoff in performance associated with investments in IC development, the unpredictability of new algorithm discovery makes investing in signal processing capability somewhat more risky. However, the higher risk of algorithm investment is more than offset by its most fundamental payoff: the development of entirely new capabilities resulting from the application of new concepts and mathematics. 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. In contrast, breakthroughs in functionality add entirely new tools to the signal processing toolbox, and may change the entire nature of the signal processing problem and implementation complexity.

With regard to new algorithm concepts and capabilities, the scope of research opportunities 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 focusing on knowledge-based and cognitive systems opens up to scrutiny entirely new types of both functionality and computational complexity.

The investment needed to maintain, and even expand, an active and robust embedded processing research community is a fraction of the investment needed to keep semiconductors on the Moore's Law growth curve, primarily because algorithm research costs are dominated by salaries, which rise at a much slower rate than the cost of new fabrication facilities. Therefore, relatively modest increases in algorithm and software research are sufficient to maintain the total progress in signal processing performance. In particular, as we move into a future where the certainty of Moore's Law hardware improvements is in question, and the investment gap between algorithm research and improved semiconductor fabrication facilities continues to widen, increasing our investment in algorithm research appears to be an attractive and proven means to sustaining exponential growth in embedded application performance.