Energy Scalable System Design

Amit Sinha, Alice Wang, and Anantha Chandrakasan

***Abstract*—We introduce the notion of energy-scalable system-design. The principal idea is to maximize computational quality for a given energy constraint at all levels of the system hierarchy. The desirable energy-quality (E–Q) characteristics of systems are discussed. E–Q behavior of algorithms is considered and transforms that significantly improve scalability are analyzed using three distinct categories of commonly used signal-processing algorithms on the StrongARM SA-1100 processor as examples (*viz.*, filtering, frequency domain transforms and classification). Scalability hooks in hardware are analyzed using similar examples on the Pentium III processor and a scalable programming methodology is proposed. Design techniques for true energy scalable hardware are also demonstrated using filtering as an example.**

***Index Terms*—Algorithmic transforms, energy scalable, low power, variable precision.**

I. Introduction

IN EMBEDDED systems, energy is a precious resource and must be used efficiently. Therefore, it is highly desirable that we structure our algorithms and systems in such a fashion that computational accuracy can be traded off with energy requirement. At the heart of such transformations lies the concept of incremental refinement [1]. Consider a scenario where an individual is using his laptop for a video telephone application. Based on the current battery state and overall power-consumption model [2] the system should be able to predict its uptime. If the battery life is insufficient, the user might choose to tradeoff some quality/performance and extend the battery life of his laptop. Consider another scenario where a distributed sensor network [3] is being used to monitor seismic activity from a remote basestation. Sensor nodes are energy constrained and have a finite lifetime. It would be highly desirable to have energy scalable algorithms and protocols running on the sensor network. The remote basestation should have the capability to dynamically reduce energy consumption (to prolong mission lifetime if uninteresting events have occurred) by altering the throughput and computation accuracy. This type of behavior necessitates system redesign so that every computational step leads us incrementally closer to the output.

Manuscript received December 9, 2000; revised March 13, 2002. This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) and in part by the Air Force Research Laboratory, Air Force Material Command, USAF, under Agreement F30602-00-2-0551.

A. Sinha was with the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology, Cambridge, MA 02139 USA. He is now with Engim, Incorporated, Acton, MA 01720 USA (e-mail: sinha@engim.com).

A. Wang and A. Chandrakasan are with the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology (MIT), Cambridge, MA 02139 USA (e-mail: aliwang@mtl.mit.edu; anantha@mtl.mit.edu).

Publisher Item Identifier S 1063-8210(02)03189-X.

Energy–quality (E–Q) tradeoffs have been explored in the context of encryption processors [4]. A large class of systems, as they stand, do not render themselves to such E–Q scaling. With hardware hooks and simple algorithmic modifications, the E–Q behavior of the system can be modified such that if the available computational energy is reduced, the proportional hit in quality is minimal. However, one must ensure that the energy overhead attributed to the improved scalability is insignificant compared to the total energy consumption. It may be possible to do a significant amount of preprocessing such that the E–Q behavior is close to perfect but we might end up with a situation where the overall energy consumption is higher compared to the unscalable system. This defeats the basic idea behind having a scalable system *viz.* overall energy efficiency.

II. Energy Quality Scalability

We now formalize the notion of a desirable E–Q behavior of a system. The E–Q graph of an algorithm is the function $Q(E)$, representing some quality metric (e.g., mean-square error, peak signal-to-noise ratio, etc.) as a function of the computational energy. There may exist situations where the notion of a quality metric is unclear. However, in this paper we are dealing primarily with signal processing systems where the notion of a quality metric is usually unambiguous. Consider two systems (I and II) that perform the same function. Ideally, from an energy perspective, II would be a more efficient scalable system compared to I if

$$Q_{II}(E) > Q_I(E), \qquad \forall E. \tag{1}$$

In most practical cases, (1) will not hold over all energy values. There might be a preprocessing overhead as a result of which the maximum energy consumptions might be different for the two cases (i.e., $E_{\max,II} > E_{\max,I}$). Nevertheless, as long as the (1) holds over a significant range of computational energies, overall efficiency is assured. Let us assume that there exists a quality distribution $p_Q(x)$, i.e., from system statistics we are able to conclude that the probability that we would want a quality x is $p_Q(x)$. A typical quality distribution is shown in Fig. 1. The average energy consumption per output sample can then be expressed as

$$\overline{E} = \int p_Q(x)E(x)\,dx \tag{2}$$

where $E(Q)$ is the inverse of $Q(E)$. When the quality distribution is unknown, we would like the E–Q behavior to be maximally concave downwards (with respect to the energy axis)

$$\frac{d^2Q(E)}{dE^2} \leq 0. \tag{3}$$

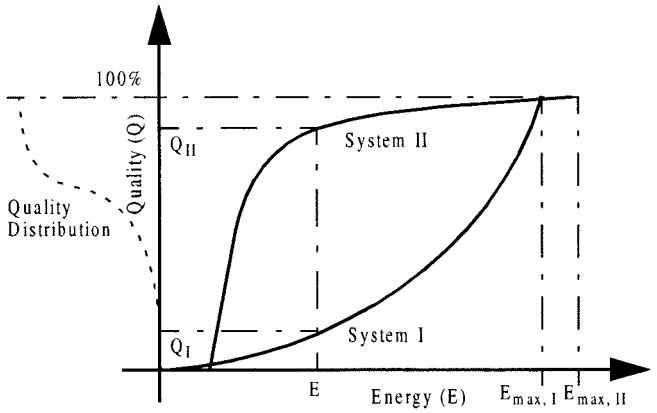

Fig. 1. E–Q formal notions.

The E–Q behavior suggested by (3) is not always attainable globally, i.e., across $0 \leq E \leq E_{\max}$ as we will see subsequently. However, on an average case, for a given energy availability, E, we would like the obtainable quality $Q(E)$ to be as high as possible.

III. Energy Scalable Application Design

Consider the simple power series shown in (4). Such power series are frequently encountered in Taylor expansions used to evaluate transcendental functions

$$y = f(x) = 1 + k_1 x + k_2 x^2 + \cdots + k_N x^N. \tag{4}$$

A standard implementation of the algorithm would have an N-step loop that would multiply the current value of the computed power of x with x and accumulate the result in y. Let us assume we have to compute $f(2)$ for $N = 100$. If the k_is are similar, even after $N - 1$ steps in the loop, the value accumulated in y would be approximately 50% off from the final value since $2^N/f(2) \approx 1/2$. In terms of E–Q performance, the algorithm does not do well. Assuming that the amount of energy required to evaluate $f(2)$ on a processor is $E_{\max}$ and that each step dissipates the same amount of energy (ignoring interinstruction effects, etc.), we have about 50% computational accuracy after dissipating $(1 - 1/N)E_{\max}$ energy. However, if we had to evaluate $f(0.5)$, the most significant terms would occur in the first few steps in the loop and the E–Q behavior would be better. Based on the above analysis, we can conclude that transforming the algorithm, as shown in Table I, will result in the most significant computations occurring early in the loop as a result of which, the computational energy could be reduced, without taking a significant hit in accuracy.

Fig. 2 shows the E–Q graphs for the original and modified power-series algorithm. It captures the all the basic ideas. i) E–Q behavior is, in general, data dependent. It is possible to come up with pathological cases where the transformed algorithm would have a E–Q behavior very close to the original. However, from an energy efficiency perspective, it is the average E–Q performance that matters. ii) It is desirable to have an E–Q graph above the baseline ($E = Q$ on a normalized scale). This would imply that the marginal return in accuracy from successive units of computational energy is diminishing. Therefore, if the available energy is reduced by 10%, the quality degradation is less that 10%, the lesser, the better. iii) There is an energy overhead associated with the transform that should be insignificant compared to the total energy.

A. Filtering Application

Finite impulse response (FIR) filtering is one of the most commonly used digital signal processing (DSP) operations. FIR filtering involves the inner product of two vectors, one of which is fixed and known as the impulse response $h[n]$ of the filter [6]. An N-tap FIR filter is defined by (5)

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k]. \tag{5}$$

Various low power and energy efficient implementations of the FIR filter have been proposed and implemented [7]. The approximate processing techniques proposed in [8] reduce the total switched capacitance by dynamically varying the filter order based on signal statistics.

However, when we analyze the FIR filtering operation from a pure inner product perspective, it simply involves N multiply and accumulate (MAC) cycles. For desired E–Q behavior, the MAC cycles that contribute most significantly to the output $y[n]$ should be done first. Each of the partial sums, $h[k]x[n-k]$, depends on the data sample and therefore it is not apparent which ones should be accumulated first. Intuitively, the partial sums that are larger in magnitude (and can therefore affect the final result significantly) should be accumulated first. Most FIR filter coefficients have a few coefficients that are large in magnitude and progressively reduce in amplitude. Therefore, a simple but effective most-significant-first transform involves sorting the impulse response in decreasing order of magnitude and reordering the MACs such that the partial sum corresponding to the largest coefficient is accumulated first as shown in Fig. 3.

Undoubtedly, the data sample multiplied to the coefficient might be so small as to mitigate the effect of the partial sum. Nevertheless, on an average case, the coefficient reordering by magnitude yields a better E–Q performance than the original scheme. Fig. 4 illustrates the scalability results for a low-pass filtering of speech data sampled at 10 kHz using a 128-tap FIR filter whose impulse response (magnitude) is also outlined. The average energy consumption per output sample (measured on the StrongARM SA-1100 [5] operating at 1.5 V power supply and 206 MHz frequency) in the original scheme is 5.12 mJ. Since the initial coefficients are not the ones with most significant magnitudes the E–Q behavior is poor. Sorting the coefficients and using a level of indirection (in software that amounts to having an index array of the same size as the coefficient array), the E–Q behavior can be substantially improved. It can be seen that fluctuations in data can lead to deviations from the ideal behavior suggested by (3), nonetheless, overall concavity is still apparent. The energy overhead associated with using a level of indirection on the SA-1100 was only 0.21 mJ that is about 4% of the total energy consumption. Fig. 5 shows the ratio of the energy consumed in the unsorted system to the sorted system for a given quality.

TABLE I
POWER SERIES COMPUTATION

Original Algorithm	Transformed Algorithm

`xpowi = 1.0; y = 1.0;`
`for( i=1; i<N; i++ ) {`
`xpowi *= x;`
`y += xpowi*k[i];`
`}`
	`if( x>1.0 ) {`
	`xpowi = pow(x,N);`
	`y = k[N]*xpowi+1;`
	`for( i=N-1; i>0; i-- ) {`
	`xpowi /= x;`
	`y += xpowi*k[i]; }`
	`}`
	`else { // original algo`
	`}`

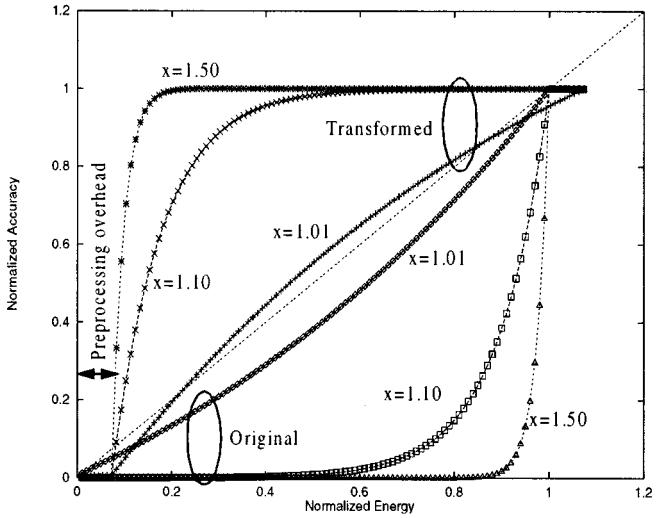

Fig. 2. E–Q performance of power series algorithm.

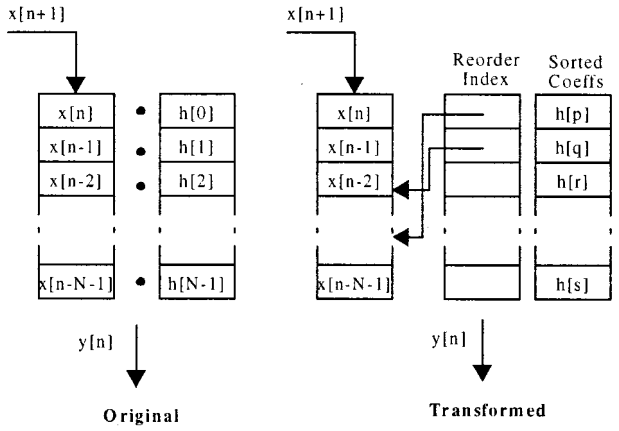

Fig. 3. FIR filtering with coefficient reordering.

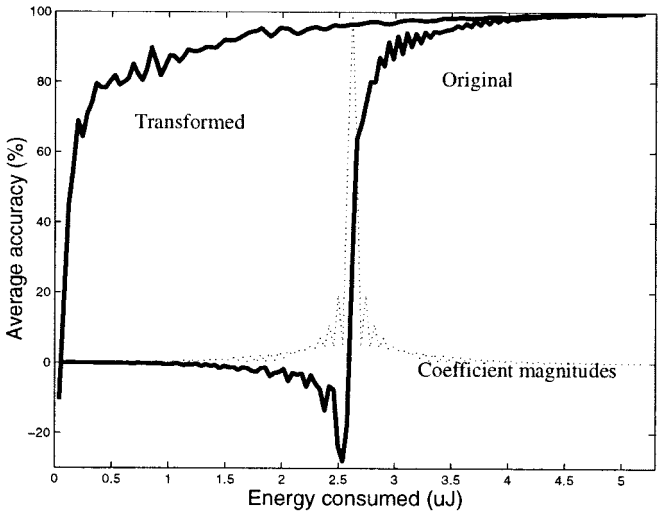

Fig. 4. E–Q graph for original and transformed FIR filtering.

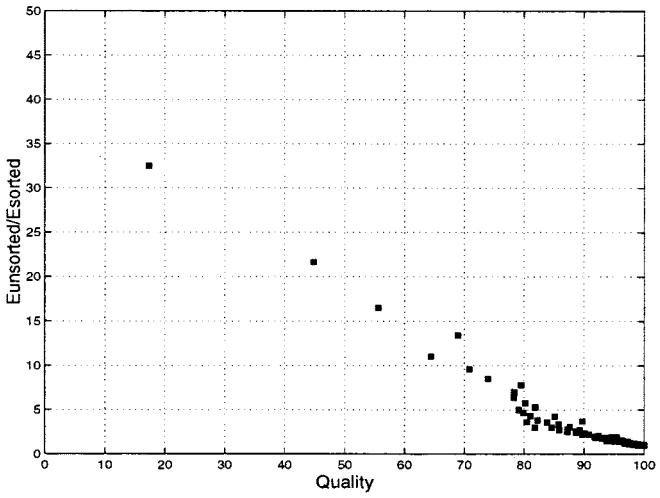

Fig. 5. Energy inefficiency of unsorted system compared to the sorted case.

In FIR filtering, the input data samples are unknown *a priori*. The partial sum that is most significant is not completely deterministic until all of them have been computed. More sophisticated schemes could involve sorting both the data samples and the coefficients and using two levels of indirection to perform the correct inner product first by picking up the partial sum corresponding to the largest coefficient, then the one corresponding to the largest data sample and so on. The overhead associated with such a scheme involves real time sorting of incoming samples. Assuming that we have a presorted data array at time n, the next data sample $x[n+1]$ can be inserted into the right position using a binary search type technique which can be done in $O(\log N)$. The scalability gains might not be substantial compared to the simpler scheme discussed before. However, in applications such as autocorrelation that involves an inner product of a data stream with a shifted version of itself, sorting both the vectors in the inner product would yield significant improvements in E–Q behavior.

B. Image Decoding Application

The discrete cosine transform (DCT), which involves decomposing a set of image samples into a scaled set of discrete cosine basis functions, and the inverse discrete cosine transform (IDCT), which involves reconstructing the samples from the basis functions, are crucial steps in digital video [9]. The 64-point, 2-D DCT and IDCT (used on 8×8 pixel blocks of an image) are defined respectively as

$$X[u, v] = \frac{c[u]c[v]}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} x[i, j] \cos\left(\frac{(2i+1)u\pi}{16}\right) \cdot \cos\left(\frac{(2j+1)v\pi}{16}\right) \tag{6}$$

$$x[i, j] = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} c[u]c[v]X[u, v] \cos\left(\frac{(2i+1)u\pi}{16}\right) \cdot \cos\left(\frac{(2j+1)v\pi}{16}\right). \tag{7}$$

DCT is able to capture the spatial redundancy present in an image and the coefficients obtained are quantized and compressed. Most existing algorithms attempt to minimize the number of arithmetic operations (multiplications and additions) usually relying on the symmetry properties of the cosine basis functions (similar to the fast Fourier transform (FFT) algorithm) and on matrix factorizations [10]. The E–Q behavior of these algorithms is not good as they have been designed such that computation takes a minimal yet constant number of operations. The forward mapping-IDCT (FM-IDCT) algorithm, proposed in [11], can be shown to have an E–Q performance that is much better than other algorithms. The algorithm is formulated as follows:

$$\begin{bmatrix} x_{0,0} \\ x_{0,1} \\ x_{0,2} \\ \vdots \\ \vdots \\ x_{7,7} \end{bmatrix} = X_{0,0} \begin{bmatrix} c_0^{0,0} \\ c_1^{0,0} \\ c_2^{0,0} \\ \vdots \\ \vdots \\ c_{63}^{0,0} \end{bmatrix} + X_{0,1} \begin{bmatrix} c_0^{0,1} \\ c_1^{0,1} \\ c_2^{0,1} \\ \vdots \\ \vdots \\ c_{63}^{0,1} \end{bmatrix} + \cdots + X_{7,7} \begin{bmatrix} c_0^{7,7} \\ c_1^{7,7} \\ c_2^{7,7} \\ \vdots \\ \vdots \\ c_{63}^{7,7} \end{bmatrix} \tag{8}$$

where, $x_{i,j}$ are the reconstructed pels, X_{ij} are the input DCT coefficients, and $[c_k^{i,j}]$ is the 64×64 constant reconstruction kernel.

The improved E–Q behavior of the FM-IDCT algorithm can be attributed to the fact that most of the signal energy is concentrated in the DC coefficient ($X_{0,0}$) and in general, in the low-frequency coefficients as shown in Fig. 7. Instead of reconstructing each pixel by summing up all its frequency contributions, the algorithm incrementally accumulates the entire image based on spectral contributions from the low to high frequencies. Figs. 6 and 8 illustrate the E–Q behavior of the FM-IDCT algorithm. It is obvious from Fig. 8 that almost 90% image quality can be obtained from as little as 25% of the total energy consumption.

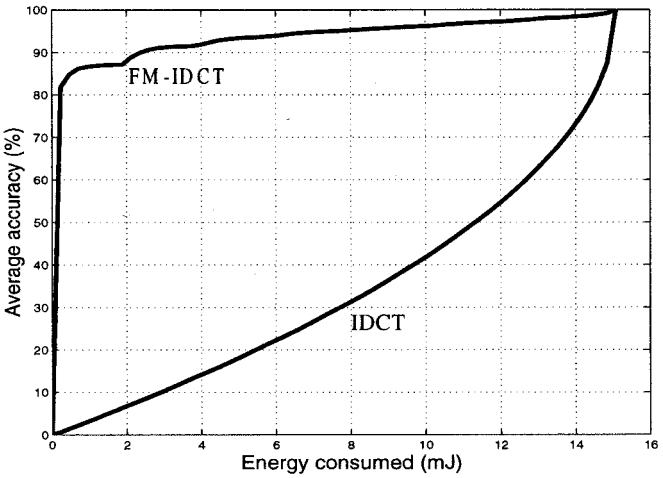

Fig. 6. E–Q graph for FM-IDCT versus normal IDCT.

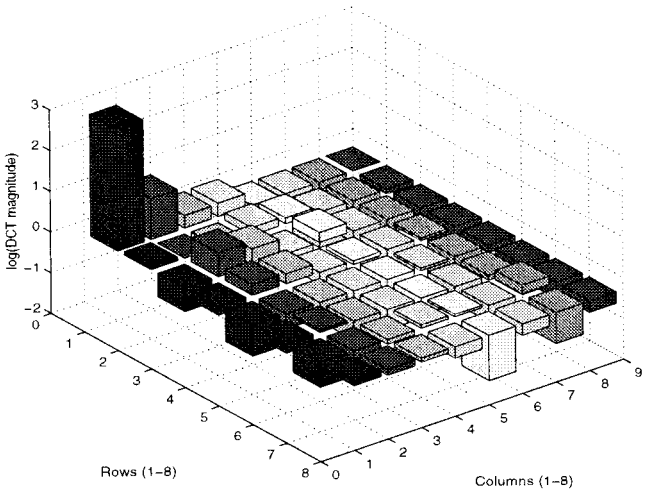

Fig. 7. 8×8 DCT coefficient magnitudes averaged over a sample image.

In terms of the overhead requirement, the only change that is required is that we now need to store the IDCT coefficients in a transposed fashion (i.e., all the low-frequency components first and so on).

C. Classification Using Beamforming

Beamforming algorithms can be used to aggregate highly correlated data from multiple sensors into one representative signal. The advantage of beamforming is twofold. First, beamforming is used to enhance the desired signal while interference or uncorrelated sensor noise is reduced. This leads to an improvement in detection and classification of the target. Second, beamforming reduces redundant data through compression of multiple sensor data into one signal. Fig. 9 shows a block diagram of a wireless network of M sensors utilizing beamforming for local data aggregation.

We have studied various beamforming algorithms that fall under the category of "blind beamforming" [12]. These beamformers provide suitable weighting functions $w_i[n]$ to satisfy a given optimality criterion, without knowledge of the sensor locations. In this paper, we will show energy scalability for one particular blind beamforming algorithm, the least mean

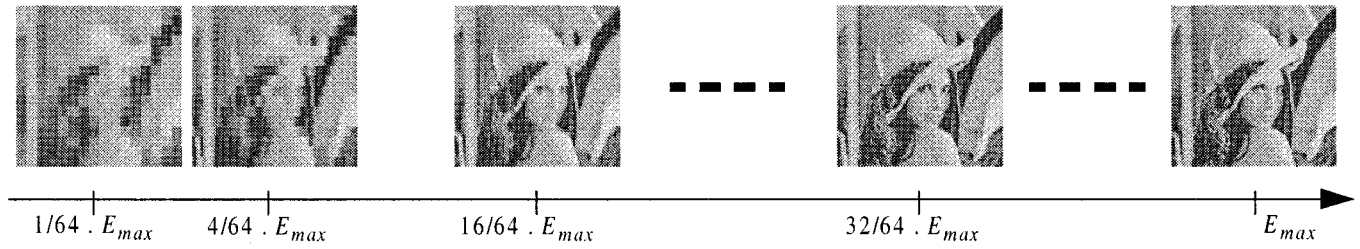

Fig. 8. Illustrating the incremental refinement property with respect to computational energy of the FM-IDCT algorithm.

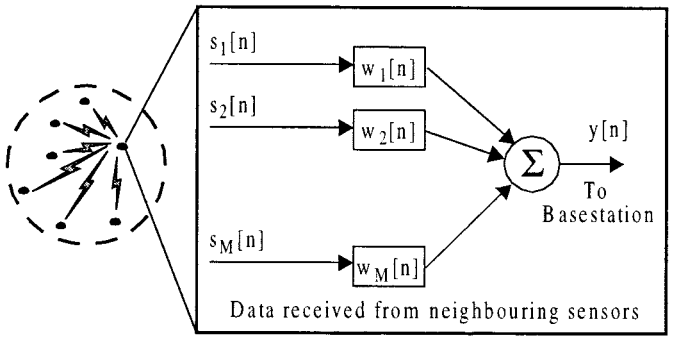

Fig. 9. Beamforming for data aggregation.

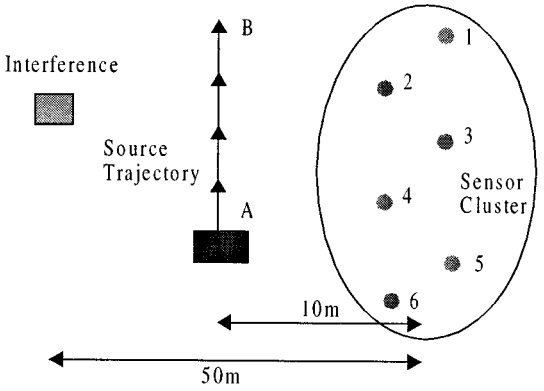

Fig. 10. Sensor testbed.

squares (LMS) beamforming algorithm. The LMS algorithm uses a minimum mean squared error criterion to determine the appropriate array weighting filters. This algorithm is considered an optimum algorithm, and is highly suitable for power aware wireless sensor networks [13].

We will now show how algorithmic transformations can be used to improve the E–Q model for LMS beamforming. Fig. 10 shows our testbed of sensors for this example. We have an array of six sensors spaced at approximately 10 meters, a source at a distance of 10 meters from the sensor cluster, and interference at a distance of 50 meters. We want to perform beamforming on the sensor data, measure the energy dissipated on the StrongARM SA-1100, calculate the matched filter output (quality), and provide a reliable model of the E–Q relationship as we vary the number of sensors in beamforming.

In Scenario 1, we will perform beamforming without any knowledge of the source location in relation to the sensors. Beamforming will be done in a pre-set order $\langle 1, 2, 3, 4, 5, 6\rangle$. The parameter we will use to scale energy is n, the number of sensors in beamforming. As n is increased from 1 to 6, there is a proportional increase of energy. As the sensor moves from location A to B we take snapshots of the E–Q curve, shown in Fig. 11. This curve shows that with a preset beamforming order, there can be vastly different E–Q curves, which leads to a very poor energy-quality model. When the source is at location A, the beamforming quality is only at maximum when sensors 5 and 6 are beamformed. Conversely, when the source is at location B, the beamforming quality is close to maximum after beamforming two sensors. Therefore, for this setup, since the E–Q curve is highly data dependent, an accurate E–Q model for LMS beamforming is not possible.

An intelligent alternative is to perform some initial pre-processing of the sensor data to determine the desired beamforming

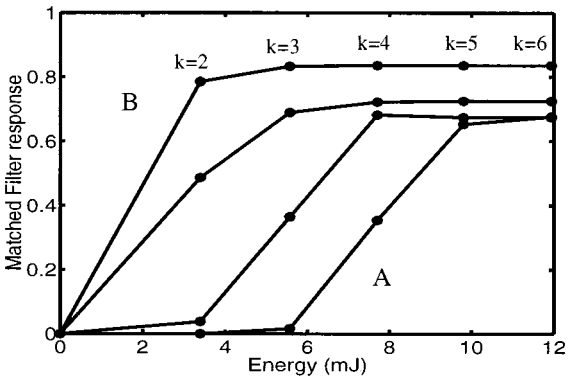

Fig. 11. E–Q snapshot for Scenario 1.

order for a given set of sensor data. Intuitively, we want to beamform the data from sensors that have higher signal energy to interference energy. Using the most-significant-first transform, which was proposed earlier, the E–Q scalability of the system can be improved. To find the desired beamforming order, first the sensor data energy is estimated. Then the sensor energies are sorted using a quicksort method. The quicksort output determines the desired beamforming order. Fig. 12 shows a block diagram of the transformed system.

In Scenario 2, we apply the most-significant-first transform to improve the E–Q curves for LMS beamforming. Fig. 13 shows the E–Q relationship as the source moves from location A to B. In this scenario, we can ensure that the E–Q graph be monotonically increasing, thus improving our E–Q models. However, there is a price to pay in computation energy. If the energy cost required to compute the correlation and quicksort was large compared to LMS beamforming, then the extra scalability is not

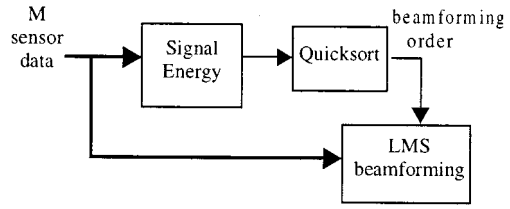

Fig. 12. "Sort by significance" preprocessing.

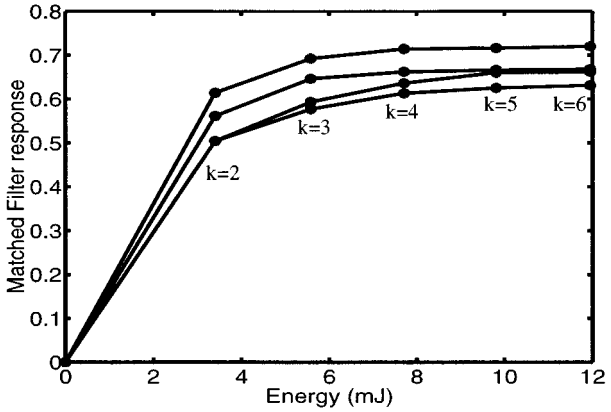

Fig. 13. E–Q snapshot for Scenario 2.

worth the effort. However, in this case, the extra computational cost was 8.8 mJ of energy and this overhead is only 0.44% of the total energy for LMS beamforming (for the two-sensor case).

IV. Exploiting Hardward Hooks for Scalability

In many multimedia and signal processing applications, the bit precision used for data representation is variable depending on the amount of accuracy sought. The number of bits used to represent data directly translates to the number of quantization levels that the dynamic range could be divided into and determines the quantization error. The variance of the quantization noise in uniform quantization, for example, is given by $\sigma_Q^2 = \Delta^2/12$, where Δ is the step-size [14]. The higher the bit precision, the greater is the accuracy.

However, when algorithms are implemented on standard DSP or general-purpose processors, computational precision is fixed. Although compilers support primitive integer datatypes (e.g., char, int, long) that allow some flexibility to the programmer in choosing the container/variable size, computation is essentially done on a fixed width datapath (e.g., 32-bit wide on a 32-bit machine). This is definitely wasteful in terms of power as unnecessary sign extension bits are toggling. Further, meeting a particular performance requirement on a wider datapath requires a higher voltage to reduce the individual gate delays on the longer critical path. Ideally, we would like to dynamically alter the datapath width to match the computational requirement. We would also like to tradeoff computational quality with power savings for energy scalable applications. The power savings obtained from variable precision computation can be substantial especially if used in conjugation with dynamic voltage scaling (DVS). Consider a scenario where a distributed sensor network has been deployed to gather acoustic data involving 16-bit processing. Based on battery energy availability, the remote basestation might want to extend the mission life at the cost of reduced accuracy. Switching to 8-bit mode can result in theoretically 87.5% processor power reduction. In addition, if the A/D circuits are scalable, lower resolution can result in significant overall system power reduction. Fig. 14 shows a variable precision microprocessor system that delivers fixed performance but can vary the power consumption based on precision requirement on the application. The system has DVS capability and the processor supports a variable bit-width datapath. When lower precision is required, the processor lowers the voltage and frequency based on a lookup table. It also switches to the appropriate code.

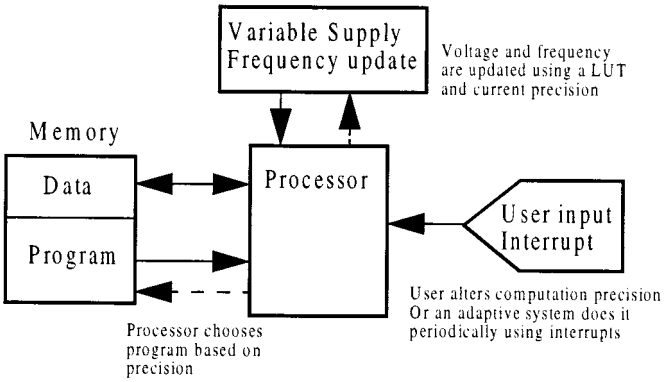

Fig. 14. Variable precision microprocessor system.

Reconfigurable architectures have been proposed in literature to tune hardware in accordance with the applications precision requirement. Bit-serial computation is another effective way to perform variable precision arithmetic. In the following section, a variable voltage-adaptive-precision FIR filter implementation is described. Although, ASICs can perform variable precision computation at a fine granularity, general-purpose processors allow flexibility and development ease at the cost of coarser granularity. In this section, we demonstrate how variable precision computation can be accomplished at a coarse granularity on the Pentium III.

A. Variable Precision Computation on the Pentium III

Variable precision computation is impossible unless hooks are provided in the processor. The Intel Pentium MMX technology [16] can be exploited for variable-precision computation. It defines new 64-bit datatypes as shown in Fig. 15. The 8 MMX registers do not add to the processor state but are aliased to the floating point registers. MMX instructions move packed datatypes (bytes, words, doublewords) and the quadword to and from memory and the general-purpose registers. However, when performing arithmetic and logical operations on packed datatypes, the MMX instructions operate in parallel on individual elements. MMX supports signed and unsigned arithmetic in saturation and wraparound modes.

B. Software Approach

We would like to have a programming mechanism that helps us exploit all MMX features without significant change in intuitiveness or high-level programming style. This can be achieved by using two powerful C++ features called templates and operator overloading [15]. A function template enables

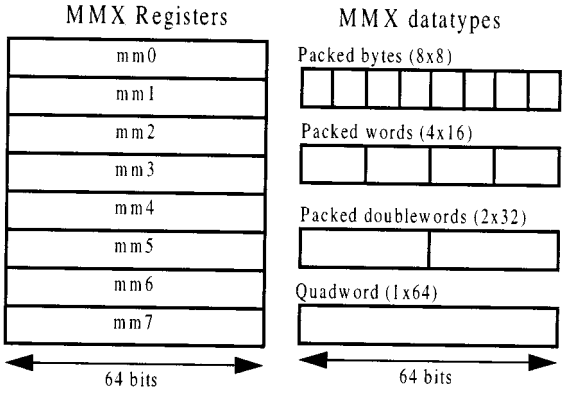

Fig. 15. Pentium III MMX registers and data types.

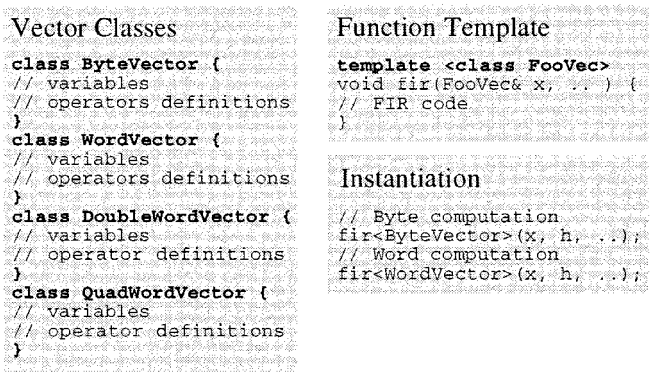

Fig. 16. Vector class implementation using SIMD arithmetic.

the programmer to write code with symbolic datatypes/classes, providing a mechanism to preserve the semantics of function definitions and calls, without having to bypass C++'s strong type checking. The programmer parameterizes all or a subset of the types in the interface and the compiler automatically generates particular instances of the function by varying the type. Fig. 16 shows the four classes describing byte, word, doubleword and quadword vectors, respectively. Our approach to coding is to use a "MATLAB" style of programming. Essentially, parallel code is best captured in vector space. We define the vector classes with an array of the data type and a size parameter as a data structure. All vector operations use overloaded operators. These overloaded operators enable intuitive vector style programming, e.g., $a = b + c$ can be written, where a, b, and c are equal length vectors (with bytes, word, doubleword or quadword as elements). The implementation of overloaded operator functions use MMX instruction intrinsics that allow parallel operation. The function template mechanism allows the programmer to write a single instance of the application. Upon invocation with the appropriate vector class the required code is automatically generated.

One of the hardware limitations of the MMX technology is that not all operations are supported for all the packed datatypes. Packed multiplication can only be performed on pairs of 4-word operands. Users may choose the lower-order or the higher order 16-bits of the intermediate 32-bit result.

V. Energy Scalable Hardware Design

In the last section, we demonstrated how certain features on standard processors could be exploited for energy scalability. The more the hooks available in hardware, the better the E–Q performance. In this section, we demonstrate the design of a completely scalable datapath using FIR filtering as an example.

There exists a wealth of techniques proposed in literature for low power implementations of FIR filters. When the filter coefficients are fixed, the flexibility offered by a dedicated multiplier is not required. Distributed Arithmetic (DA) is a bit-serial, multiplier-less technique that exploits the fact that one of the vectors in the inner product is fixed [17]. All possible intermediate computations (for the fixed vector) are stored in a lookup table (LUT) and bit slices of the variable vector are used as addresses for the LUT. A four-tap DA-based FIR filter is shown in

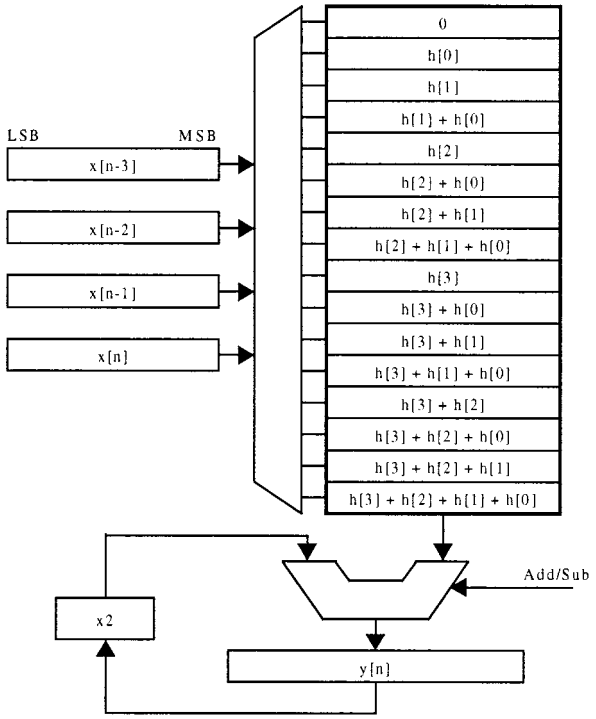

Fig. 17. DA implementation of a four-tap filter.

Fig. 17. In general, an N tap filter requires an LUT of size 2^N. Area-delay tradeoffs have been explored in [18]. For instance, the LUT of size 2^N can be split up into two different LUTs of size $2^{N/2}$ at the cost of an additional adder. The output $y[n]$ is not available every cycle. If the bit precision of the data samples $x[n-k]$ is M, then a valid output $y[n]$ is available every M cycles. In the MSB first implementation of Fig. 17, it has been shown in [19] that each successive intermediate value is closer to the final value in a stochastic sense.

If we assume that the energy consumption per cycle is E_0, it is straightforward to see that the energy consumption per output sample is ME_0 (i.e., the energy consumption is linearly dependent on the bit precision of data samples). The filter must be designed so that it can accommodate the largest possible input signals (i.e., those requiring the maximum possible bits for representation). In most signal processing applications, the signals we get are correlated and only few samples actually require the

maximum bit precision for representation. The average bit precision required by typical 16-bit speech data is around 6.9 bits. Most filtering circuits will be designed to accommodate 16-bit data and in our DA based implementation, with fixed precision, the energy required per output sample would be $16E_0$.

Let us assume that the maximum precision requirement is $M_{\max}$ and the immediate precision requirement is $M \leq M_{\max}$. This should scale down the energy per output sample by a factor $M/M_{\max}$. We should also exploit the fact that lesser precision implies that the same computation can be done faster (i.e., in M cycles instead of $M_{\max}$). This can be done by switching down the operating voltage such that we still meet the worst case throughput requirement (i.e., corresponding to one output sample every $M_{\max}$ cycles when operating at $V_{\max}$) while obtaining quadratic energy savings.

A. Filter Architecture

DA is particularly suited to variable precision filtering. The filter that we have implemented is an eight-tap low-pass filter. Both data and coefficients use a 16-bit, two's complement representation. The LUT has two^8 entries. There are eight-data registers (corresponding to the eight-data samples that are required for every output sample). To implement a precision-on-demand scheme, we need to determine the minimum precision that is required to compute the output without any loss in accuracy.

Consider the two cases shown in Fig. 18(a) and (b). In both cases, Fig. 18(a) and (b), we have four registers, each with two's complement 16-bit data. The sign extension bits have been shaded in every register. Notice that in case of small numbers the MSBs are "0" for positive quantities and "1" for negative quantities. No accuracy is lost if we reject the sign-extension bits. Determining the number of sign-extension bits is relatively simple in hardware and can be done by the circuit shown in Fig. 18(c) [20]. The "1" outputs of the "mask" determine the sign-extension bits in each register. The number of sign-extension bits that can be rejected is equal to the minimum of the sign-extension bits among all the registers. This can be obtained by simply "anding" all the individual mask outputs from each register. In our DA implementation, the final mask (obtained by "anding" all the individual masks) determines the number of cycles, M, required for computing the current output. For a four-tap implementation, with the register contents as shown in Fig. 18(b), the number of cycles that will be required is $M = 6$, instead of $M_{\max} = 16$. The energy overhead due to the sign-extension hardware is small since the duty cycle is low (one per output sample).

Once the precision requirement for a sample has been determined we know the number of cycles M required for obtaining the result $y[n]$. In general, the filter will have to be designed such that it meets a fixed throughput requirement in the worst case operating precision, i.e., we have one output sample $y[n]$ at least every $M_{\max}$ cycles. But with a precision requirement $M \leq M_{\max}$, we would have a valid output earlier. Since, it does not pay to do computations any faster than required, we would have to idle for $(M_{\max} - M)$ cycles.

We can exploit this fact, that we have more time for computation, by lowering the supply voltage. This results in quadratic reduction in energy and corresponding increase in computational delay. We lower the supply voltage V_{dd} from $V_{dd} = V_{\max}$ to $V_{dd} = V(M)$ such that the time required to execute M cycles at $V_{dd} = V(M)$ is equal (but not greater than) the time required to execute $M_{\max}$ cycles at $V_{dd} = V_{\max}$. This idea is summarized in Fig. 19. Implementing a scheme like this involves a variable supply voltage (a dc/dc converter) with sufficient feedback bandwidth, a variable clock-frequency generator along with the precision determining circuit in the original DA implementation. Even the best dc/dc converters do not have sufficient feedback bandwidth to track the fast changes in data precision requirement that filtering involves. Therefore, such a scheme is difficult to implement practically.

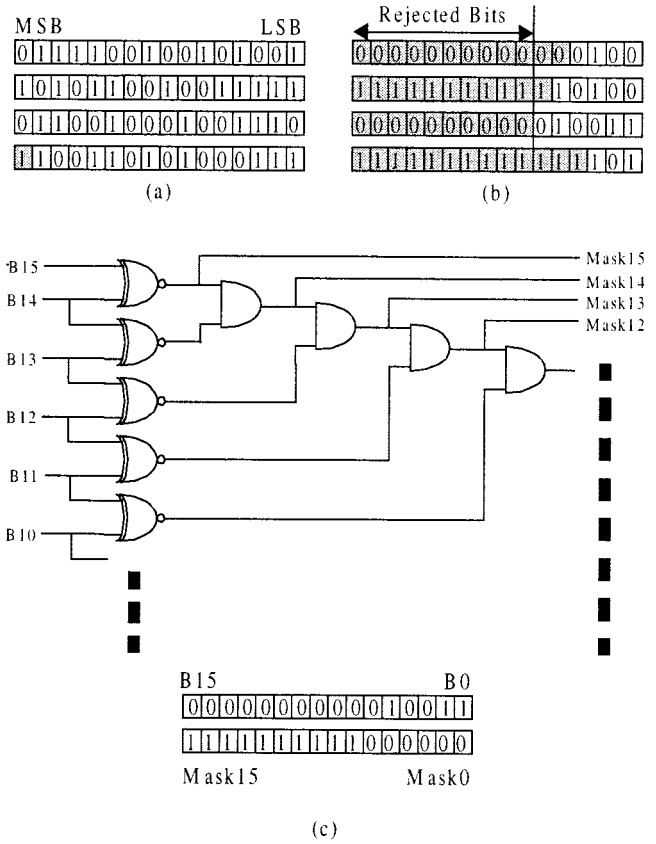

Fig. 18. Determining sign-extension bits.

To overcome the limitation posed by the dc/dc converter we implemented and tested a filtering scheme that uses two ROM and accumulator (RAC) units operating at two different supply voltages, $V_{dd} = V_{\max}$ and $V_{dd} = V_{\min}$. The RAC operating at the higher voltage is such that it can produce a valid output every $M_{\max}$ cycles and meets the throughput constraint. The RAC operating at the lower voltage is such that it can go through only $M_{\min}$ cycles before it exceeds the throughput constraint. If the current precision requirement is M ($M_{\min} \leq M \leq M_{\max}$), we execute some fraction D of $M_{\max}$ cycles, in the higher voltage RAC and the remaining cycles $(1 - D)$ of $M_{\min}$ cycles, at the lower voltage RAC. Of course, the number of cycles we spend at either voltages, must be integral and must add up to M

$$DM_{\max} + (1 - D)M_{\min} = M. \tag{9}$$

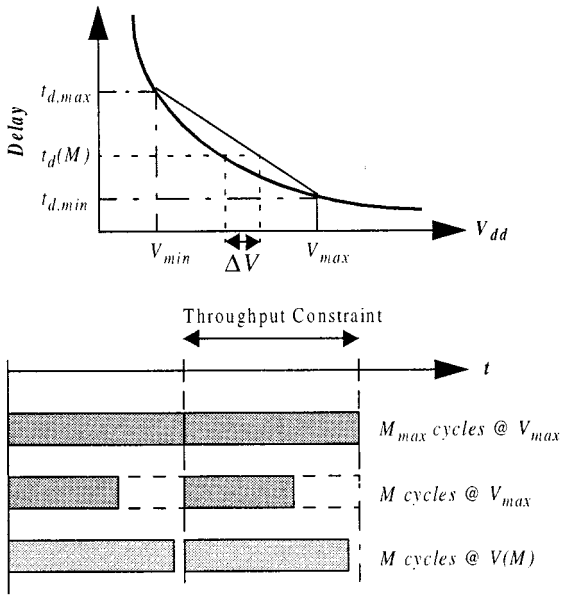

Fig. 19. Just-in-time computation.

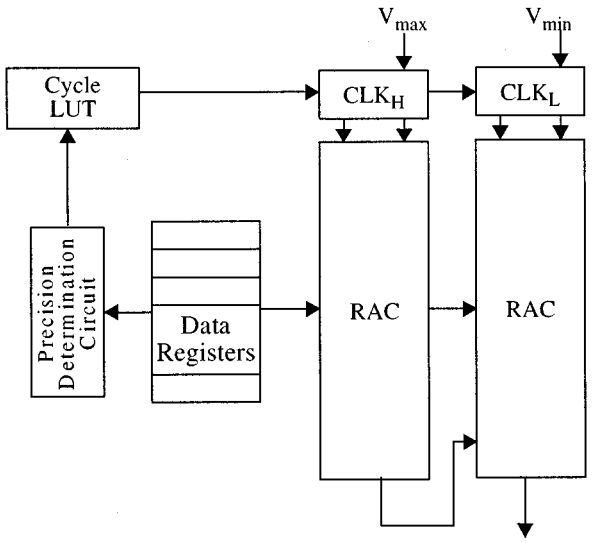

Fig. 20. Just-in-time filtering with two-supply voltages.

Fig. 20 illustrates the basic architecture of just-in-time filtering with two RAC units operating at two-supply voltages. The details of the control logic and level shifters have been omitted for clarity. The operation is very simple. When a new data sample is loaded into the register, the "precision determination circuit" computes the minimum precision required for the computation. This determines the total number of cycles M. The accumulators within each of the RACs are cleared. The "cycle LUT" then determines the number of cycles to be executed at $V_{\max}$ and $V_{\min}$, respectively, and clocks the corresponding RACs. At the end of $DM_{\max}$ cycles at $V_{\max}$, the accumulator contents of the RAC operating at $V_{\max}$ are loaded into the RAC operating at $V_{\min}$, and the RAC is clocked for the remaining cycles. The average delay per cycle in this scheme is, therefore, a simple linear combination of the delays $t_{d,\max}$ and $t_{d,\min}$. This is suboptimal compared to the arbitrary voltage scheme as shown in Fig. 19, in that we are operating at an average voltage that is higher than the optimal voltage by ΔV.

In our implementation, $M_{\min}$ was chosen to be 4. If the precision requirement was less than 4, all cycles were executed at $V_{\min}$. Therefore, once again the double-voltage scheme was suboptimal in that we did not exploit all the time we had for computation. The choice of $M_{\min}$ is data dependent. We do not want it to be very small because most of the cycles would then be spent at $V_{\max}$. At the same time, we do not want it to be large because all precision requirements below $M_{\min}$ would then be suboptimal.

B. Results

In our implementation, $V_{\max} = 2.1$ V, $V_{\min} = 1.3$ V, $M_{\max} = 16$, and $M_{\min} = 4$. Fig. 21 illustrates the precision distribution for typical speech data. Notice that the distribution peaks around $M = 4$ which implies that a large fraction of the data would require only the RAC operating at $V_{\min}$ to execute all cycles.

Fig. 21 also shows 40 000 samples of speech data. Using our variable precision filter, for the first 15 000 samples, where the data values are relatively small, the energy requirement is less than 20% of the maximum. On the other hand, the next 10 000 samples being larger in magnitude (requiring close to the maximum precision) use up a lot more energy. The average energy required for all the samples shown in Fig. 21 is about 40% of the maximum, i.e., our scheme saves about 60% energy.

Fig. 22 shows the results of our filtering scheme on 37 different speech data blocks. We can clearly see the linear variation of average energy requirement with the average precision requirement. Once again, the energy is normalized with respect to the maximum energy. For the fixed precision implementation, all the speech blocks would require almost the same energy equal to the maximum energy.

An interesting case to compare would be the classical area power tradeoff that one gets from duplicating hardware [21]. In our case that translates to having two RACs running at some intermediate voltage V_m such that each RAC is able to execute $M_{\max}$ cycles in the twice the throughput interval. So, we would have two such units running in parallel in such a fashion that we have one output sample every T_0 time. Clearly, this would result in an energy reduction by a factor $(V_m/V_{\max})^2$. The energy reduction is fixed and independent of precision requirement as shown in Fig. 22. However, if the data samples are such that the precision requirement is high, then the energy advantage of our scheme would be lost. Also plotted in Fig. 22, is the average energy variation with average precision requirement for the just the variable precision case (i.e., variable precision using single RAC, single operating voltage).

VI. Conclusion

We have introduced the notion of energy scalable systems in the context of signal processing. Such systems that render incremental refinement in quality such that the marginal returns from every additional unit of energy are diminishing are highly desirable in embedded applications. E–Q scalability enhancements

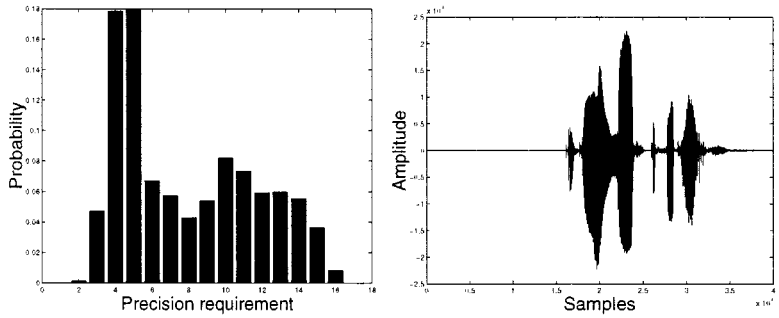

Fig. 21. Probability distribution of precision requirement and typical data.

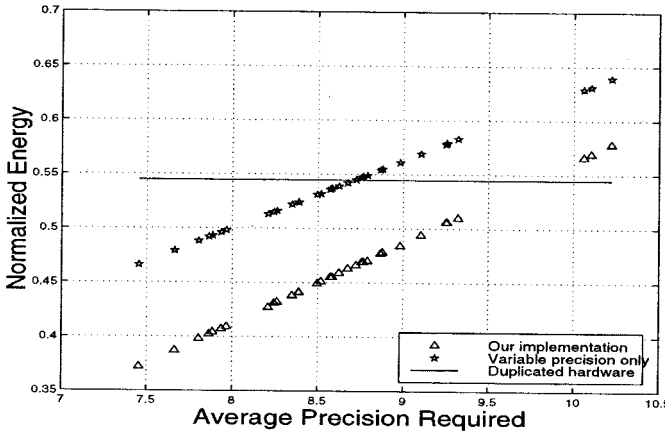

Fig. 22. Average energy as a function of average precision.

have been demonstrated from the application level down to the hardware level. Using three-broad classes of signal processing algorithms we have demonstrated that using simple transformations (with insignificant overhead) the E–Q behavior of the algorithm can be significantly improved. In general, we have shown that doing the most significant computations first enables computational energy reduction without significant hit in output quality. A programming methodology that exploits features available in most contemporary processors for E–Q scalability using variable precision computation is also described. Finally, design techniques for true energy scalable hardware are also demonstrated using filtering as an example.

REFERENCES

[1] S. H. Nawab *et al.*, "Approximate signal processing," *J. VLSI Signal Processing Syst. Signal, Image, Video Technol.*, vol. 15, no. 1/2, pp. 177–200, Jan. 1997.

[2] A. Sinha and A. Chandrakasan, "Energy aware software," in *Proc. XIII Int. Conf. VLSI Design*, Calcutta, India, Jan. 2000.

[3] A. Chandrakasan *et al.*, "Design considerations for distributed microsensor systems," in *Proc. IEEE 1999 Custom Integrated Circuits Conf.*, San Deigo, CA, May 1999, pp. 279–286.

[4] J. Goodman, A. Dancy, and A. P. Chandrakasan, "An energy/security scalable encryption processor using an embedded variable voltage DC/DC converter," *IEEE J. Solid-State Circuits*, vol. 33, pp. 1799–1809, Nov. 1998.

[5] Advanced RISC Machines Ltd., *Advance RISC Machines Architectural Reference Manual*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[6] A. V. Oppenheim and R. W. Schafer, *Discrete Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[7] A. Sinha and A. P. Chandrakasan, "Energy efficient filtering using adaptive precision and variable voltage," in *Proc. IEEE 12th Ann. ASIC Conf.*, Sept. 1999.

[8] J. T. Ludwig, S. H. Nawab, and A. Chandrakasan, "Low-power digital filtering using approximate processing," *IEEE J. Solid-State Circuits*, vol. 31, pp. 395–400, Mar. 1996.

[9] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. 23, pp. 90–93, Jan. 1974.

[10] W. H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. 25, pp. 1004–1009, Sept. 1977.

[11] L. McMillan and L. A. Westover, "A forward-mapping realization of the inverse discrete cosine transform," in *Proc. Data Compression Conf.*, Mar. 1992, pp. 219–228.

[12] Yao *et al.*, "Blind beamforming on a randomly distributed sensor array system," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 1555–1567, Oct. 1998.

[13] A. Wang, W. Heinzelman, and A. Chandrakasan, "Energy-scalable protocols for battery-operated microsensor networks," in *Proc. SIPS '99*, Oct. 1999.

[14] S. Haykin, *Communication Systems*, New York: Wiley, 1995.

[15] S. B. Lippman and J. Lajoie, *C++ Primer*, 3rd ed. Reading, MA: Addison-Wesley, 1998.

[16] Intel Architecture Optimization Reference Manual [Online]. Available: http://developer.intel.com/design/pentiumii/manuals/245 127.htm

[17] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Mag.*, pp. 4–19, July 1989.

[18] M. Mehendale, S. D. Sherlekar, and G. Venkatesh, "Area-delay tradeoff in distributed arithmetic based implementation of FIR filters," in *X Int. Conf. VLSI Design*, Jan. 1997, pp. 124–129.

[19] R. Amirtharajah, T. Xanthopoulos, and A. Chandrakasan, "Power scalable processing using distributed arithmetic," in *Int. Symp. Low Power Electronics Design*, Aug. 1999.

[20] T. Xanthopoulos, "Low power data-dependant transform video and still image coding," Ph.D. dissertation, Massachusetts Institute of Technology, Feb. 1999.

[21] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proc. IEEE*, vol. 83, pp. 498–523, Apr. 1995.

Amit Sinha received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, and the M.A. and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, in 1999 and 2001, respectively.

His research interests include low-power systems and software design with emphasis on wireless systems.

Dr. Sinha has was awarded the President of India Gold Medal from the Indian Institute of Technology, Delhi.

Alice Wang received the S.B. and M.Eng. degrees in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, in 1997 and 1998, respectively. She is currently pursuing the Ph.D. degree at the same university.

Her research interests include energy-efficient implementation of signal processing algorithms, low-power DSPs for wireless systems and subthreshold circuit design.

Prof. Wang is a Lucent Technologies Fellow.

Anantha Chandrakasan received the Ph.D. degree in electrical engineering and computer science from the University of California at Berkeley.

Currently, he is an Associate Professor of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology (MIT), Cambridge. His research interests include the energy-efficient implementation of Digital Signal Processors and Wireless Microsensor Networks.

Dr. Chandrakasan is an elected member of the Solid-State Circuits Society AdCom.