Power Scalable Processing Using Distributed Arithmetic

Rajeevan Amirtharajah, Thucydides Xanthopoulos, and Anantha Chandrakasan
Massachusetts Institute of Technology, Cambridge, MA 02139
mirth@mtl.mit.edu,duke@mtl.mit.edu,anantha@mtl.mit.edu

Abstract

A recent trend in low power design has been the employment of reduced precision processing methods for decreasing arithmetic activity and average power dissipation. Such designs can trade off power and arithmetic precision as system requirements change. This work explores the potential of Distributed Arithmetic (DA) computation structures for low power precision-on-demand computation. We present two proof-of-concept VLSI implementations whose power dissipation changes according to the precision of the computation performed.

1 Introduction and Background

A recent trend in low power design has been the demployment of reduced precision "approximate processing" methods for reducing arithmetic activity and chip average power dissipation. Such designs treat power and arithmetic precision as system parameters that can be traded-off vs. each other on and ad-hoc basis. Ludwig et. al [1] have demonstrated an approximate filtering technique which dynamically reduces the filter order based on the input data characteristics. More specifically, the number of taps of a frequency-selective FIR filter is dynamically varied based on the estimated stopband energy of the input signal. The resulting stopband energy of the output signal is always kept under a predefined threshold. This technique results in power savings of a factor of 6 for speech inputs. Larsson and Nicol [2] [3] have demonstrated an adaptive scheme for dynamically reducing the input amplitude of a Booth-encoded multiplier to the lowest acceptable precision level in an adaptive digital equalizer. Their scheme simply involves an arithmetic shift (multiplication/ division by a power of 2) of the multiplier input depending on the value of the error at the equalizer output. They report power savings of 20%.

*Prepared through collaborative participation in the Advanced Sensors Consortium sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0001. Further support was provided by DARPA.

This work explores the potential of Distributed Arithmetic (DA) [4] [5] computation structures for low power precision-on-demand computation. Distributed Arithmetic is a method of computing vector inner products without the use of a multiplier. It has a number of applications in fixed-function DSP VLSI implementations [6] [7] [8]. When used appropriately it features stochastically monotonic successive approximation properties. In this work, we present the theory behind Distributed Arithmetic and its approximate processing properties. We also present two proof-of-concept VLSI implementations, a heartbeat classifier and a DCT core processor whose power dissipation characteristics change on-the-fly according to the precision of the computation performed.

2 Distributed Arithmetic

Distributed Arithmetic (DA) [4] [5] is a bit-serial operation that computes the inner product of two vectors (one of which is a constant) in parallel. Its main advantage is the efficiency of mechanization and the fact that no multiply operations are necessary. DA has an inherent bit-serial nature, but this disadvantage can be completely hidden if the number of bits in each variable vector coefficient is equal or similar to the number of elements in each vector.

As an example of DA mechanization let us consider the computation of the following inner (dot) product of M-dimensional vectors $\mathbf{a}$ and $\mathbf{x}$, where $\mathbf{a}$ is a constant vector:

$$y = \sum_{k=0}^{M-1} a_k x_k \qquad (1)$$

Let us further assume that each vector element x_k is an N-bit two's complement binary number and can be represented as

$$x_k = -b_{k(N-1)}2^{N-1} + \sum_{n=0}^{N-2} b_{kn}2^n \qquad (2)$$

where $b_{ki} \in \{0,1\}$ is the ith bit of vector element x_k. Please note that b_{k0} is the least significant bit (LSB) of x_k and $b_{k(N-1)}$ is the sign bit.
Substituting eq. 2 in eq. 1 yields:

$$y = -\sum_{k=0}^{M-1} a_k b_{k(N-1)} 2^{N-1} + \sum_{n=0}^{N-2} [\sum_{k=0}^{M-1} a_k b_{kn}] 2^n \quad (3)$$

Let us consider the term in brackets:

$$q_n = \sum_{k=0}^{M-1} a_k b_{kn} \quad (4)$$

Because $b_{kn} \in \{0,1\}$, q_n has only 2^M possible values. Such values can be precomputed and stored in a ROM of size 2^M. The bit serial input data $(\{b_{0i}, b_{1i}, b_{2i}, \ldots, b_{ki}\}$ for $i = 0, 1, \ldots, N-1)$ is used to form the ROM address, and the ROM contents can be placed in an accumulator structure to form the outer sum of eq. 3. Successive scalings with powers of 2 can be achieved with an arithmetic shifter in the accumulator feedback path. The first term of eq. 3 $(\sum_{k=0}^{M-1} a_k b_{k(N-1)})$ is also stored in the ROM at address $\{b_{0(N-1)}, b_{1(N-1)}, b_{2(N-1)}, \ldots, b_{k(N-1)}\}$. Some extra control circuitry is necessary to ensure that the accumulator subtracts (as opposed to adding) the partial sum to the total result at sign bit time. After N cycles (as a reminder N is the bitwidth of the x_k vector elements) the final result y cas converged to its final value within the accumulator.

Figure 1 shows a detailed example of a Distributed Arithmetic computation. The structure shown computes the dot product of a 4-element vector X and a constant vector A. All 16 possible linear combinations of the constant vector elements (A_i) are stored in a ROM. The variable vector X is repackaged to form the ROM address most significant bit first. We have assumed that the X_i elements are 4-bits 2's complement (bit 3 is the sign bit.) Every clock cycle the RESULT register adds $2\times$ its previous value (reset to zero) to the current ROM contents. Moreover, each cycle the 4 registers that hold the four elements of the X vector are shifted to the right. The sign timing pulse T_s is activated when the ROM is addressed by bit 3 of the vector elements (sign). In this case the adder subtracts the current ROM contents from the accumulator state. After four cycles (bitwidth of the X_i elements) the dot product has been produced within the RESULT register.

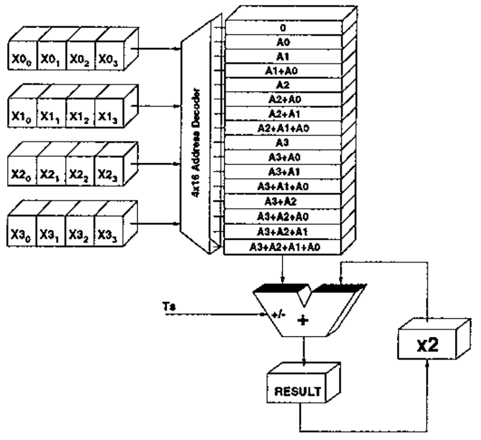

Figure 1: Distributed Arithmetic ROM and Accumulator (RAC) Structure

3 Successive Approximation Using Distributed Arithmetic

In this section, we show that when the Distributed Arithmetic operation is performed MSB first, it exhibits stochastically monotonic successive approximation properties. In other words, each successive intermediate value is closer to the final value in a stochastic sense. An analytical derivation follows:

The ith intermediate result of an MSB-first DA computation $(i > 0)$ is:

$$y_i = -q_{(N-1)} + \sum_{n=N-1-i}^{N-2} q_n 2^n \quad (5)$$

where

$$q_n = \sum_{k=0}^{M-1} a_k b_{kn} \quad (6)$$

Please note that when $i = N-1$, eq. 5 yields eq. 3.

Let us define an error term e_i, $i = 0, 1, \ldots, N-1$ as the difference between each intermediate value y_i and the final value y:

$$\begin{aligned} e_i &= y - y_i && (7) \\ e_i &= \sum_{n=0}^{N-2-i} q_n 2^n && (8) \end{aligned}$$

We model q_n as experimental values of a discrete random variable $\mathbf{q}$. The underlying stochastic experiment is random accesses of the DA coefficient ROM in the presence of random inputs. The experimental values of $\mathbf{q}$ are the DA ROM contents. The first and second order statistics of the error term e_i are:

$$\begin{aligned} E[e_i] &= E[\mathbf{q}] \sum_{n=0}^{N-2-i} 2^n && (9) \\ &= E[b_{kn}] \sum_{k=0}^{M-1} a_k \sum_{n=0}^{N-2-i} 2^n && (10) \\ &= \frac{2^{N-1-i} - 1}{2} \sum_{k=0}^{M-1} a_k && (11) \\ \sigma_{e_i}^2 &= \sigma_{\mathbf{q}}^2 (1 + 4 + \cdots + 2^{2(N-2-i)}) && (12) \\ &= \frac{2^{2(N-1-i)} - 1}{3} \sigma_{\mathbf{q}}^2 && (13) \\ &= \frac{2^{2(N-1-i)} - 1}{3} Var\left[\sum_{k=0}^{M-1} a_k b_{kn}\right] && (14) \\ &= \frac{2^{2(N-1-i)} - 1}{3} \sum_{k=0}^{M-1} a_k^2 \sigma_{b_{kn}}^2 && (15) \\ &= \frac{2^{2(N-1-i)} - 1}{12} \sum_{k=0}^{M-1} a_k^2 && (16) \end{aligned}$$

where equations 11 and 16 have been computed under the assumption that the least significant bits b_{kn} (i-large) are independent identically distributed random variables uniformly distributed between 0 and 1

$(E[b_{kn}] = 1/2,\ \sigma^2_{b_{kn}} = 1/4)$. This is a valid assumption for input DSP data [9][10]. The fact that equations 11 and 16 are monotonically decreasing functions of i (RAC cycles) shows the succesive approximation property (in probabilistic terms) of the Distributed Arithmetic mechanization.

In the next two sections we show two VLSI implementations that use the successive approximation properties described above to achieve power scalability.

4 DSP for Physiological Monitoring

An example of power scalable processing using Distributed Arithmetic is a low power DSP for physiological monitoring. The biomedical sensor is a microphone for recording heartbeats, breathing sounds, and voice data. This data will eventually be used to determine the physical condition of the wearer. The first step is detection of the heartbeats, which can be used to determine heart rate as the basis for a physiological assessment.

Evaluation of the spectrogram of the acoustic data indicates that most of the energy from heartbeat sounds lies in the low frequency range, below 200 Hz. We developed a classifier based approach to heartbeat detection that takes advantage of this spectral characteristic to improve detection performance in the presence of speech and other high frequency energy.

The basic algorithm is outlined below:

1. Preprocessing:
    - Lowpass Filtering: The data is bandlimited to below 200 Hz to eliminate as much of the voice and breath energy as possible.
    - Matched Filtering: The output of the lowpass filter is passed through a matched filter to determine the candidate heartbeat locations in the time domain.
    - Segmentation: The sensor output is divided into overlapping segments at least long enough to contain a full heartbeat in the time domain, but short enough not to contain more than one.
2. Feature Extraction: A subset of seven features is computed from the matched filter output.
3. Classification: Each feature vector is classified into a heartbeat or nonheartbeat using a parametric Gaussian multivariate classifier [11].

Assuming that the lowpass filtering occurs before sampling as an antialiasing step, the first computationally significant step is to perform the matched filtering. Matched filtering is simply convolution of the input data with a time reversed impulse response to compute a correlation output:

$$y[n] = \sum_{k=0}^{N} x[n-k]h[k] \qquad (17)$$

where $x[n]$ is the input sample at time n and $h[k]$ is the filter impulse response sample.

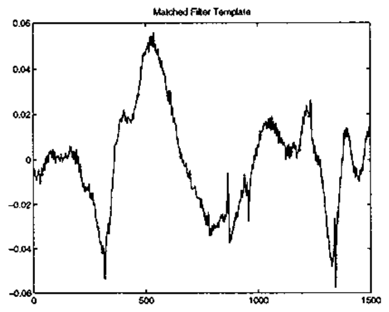

Figure 2: Matched filter template.

The architecture of the proposed sensor DSP chip follows the algorithmic architecture described above. The discrete-time matched filter is implemented using the Distributed Arithmetic Unit. Its output is then passed to a nonlinear filtering unit to calculate quantities used in segmentation. The final segmentation, feature extraction, and classification is performed by the programmable microcontroller at the end to produce the class assignment z. The buffer provides a mechanism for synchronization between the front end filtering and the backend processing. This is necessary for power reduction. The filtering front end must be running continuously to process the input samples, which arrive at a fixed rate. However, the back end classification only needs to be performed for every segment, not every input sample. The system operates as follows: first, the front end filters the input and writes important results to the buffer. A small loop is continuously executed in the microcontroller, checking to see if a full segment has been written to the buffer. The filtering units could do this, but it involves adding circuits that already exist in the ALU of the microcontroller, which is idle anyway while it is trying to detect a segment. To conserve area, we use the microcontroller rather than add complexity to the filter functional units. When a segment is detected, the microcontroller executes the feature extraction and classification code on the data in the buffer that was just written. Architectural simulation of the DSP chip using Verilog shows that 99.8% of the algorithm time is spent executing the matched filtering and other preprocessing functions. These computations therefore dominate both the time and power consumption of the algorithm.

The matched filter impulse response, or filter template, is a cleaned up version of the acoustic signature of the heartbeat. When convolved with the input data, the filter output has a large correlation peak at the time location of a heartbeat in the input. The template used in the classification algorithm is shown in Figure 2.

Figure 3 shows some typical input data (free of speech) and the matched filter output below it. There are clearly defined correlation peaks corresponding to the locations of acoustic heartbeat signatures. The segmentation phase of the preprocessing localizes the regions of this time series which have the correlation peaks. Features are then extracted from these regions and classified. The preprocessing steps, in particular

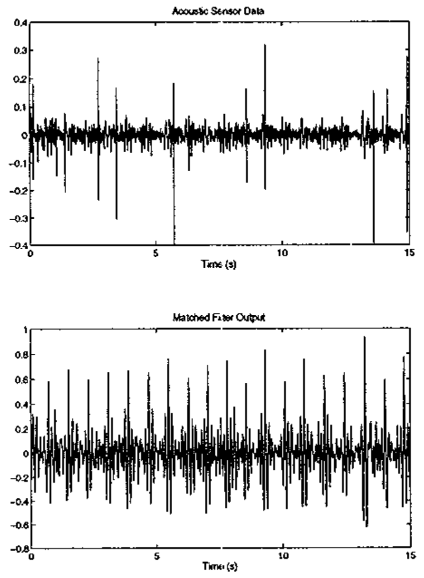

Figure 3: Matched filter output without speech.

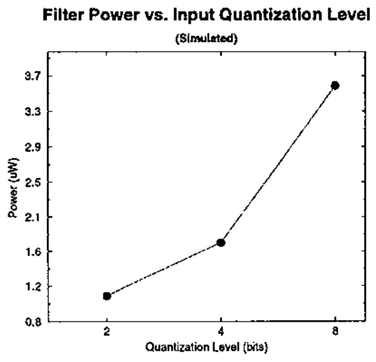

Figure 4: Power reduction versus input quantization (simulated using PowerMill).

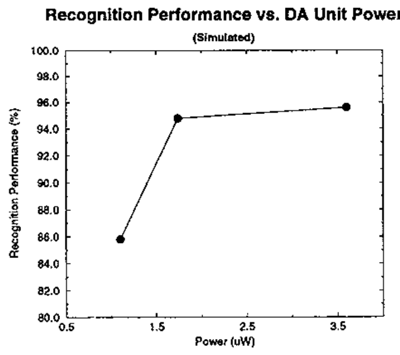

Figure 5: Recognition performance tradeoffs with DA Unit power.

the matched filtering, require the most computation. This leads to a low power DSP architecure that power optimizes these frequently performed computations.

Using Distributed Arithmetic, a complete dot product can be performed in as many cycles as correspond to the bit widths of the input samples. If the bitwidth M is less than the filter length N, this implementation requires fewer clock cycles than a multiply accumulate. This is beneficial for long filters like the matched filter described above, where $M \gg N$. The reduced clock results in low total power not just through frequency reduction, but also through increased voltage reduction since the delay constraint of the DA filter critical path is much less stringent than the multiply-accumulate architecture. Figure 4 shows the power reduction in the Distributed Arithemetic unit as the input quantization level is decreased (i.e. fewer bits of the input are shifted into the filter).

As discussed above, the bit-serial nature of the implementation also allows an alternative approach to approximate processing. By clocking the DA units at less than the full bitwidth, we are in effect reducing the input quantization level. This is roughly equivalent to injecting noise at the input of the filter. In a detection scheme like the heartbeat detection algorithm, this reduced signal to noise ratio should result in lower performance, i.e. less reliable detection of heartbeat events. However, the reduced performance has also resulted in reduced power since the switched capacitance per output filter sample decreases linearly with the number of input bits clocked in. Figure 5 shows the classifier performance reduction as the DA unit power is decreased.

5 DCT Core Processor

As another example, consider a DCT core processor. The chip architecture and circuits are described in [12], while this section focuses on the algorithmic issues in its implementation. The DCT processor has been implemented using distributed arithmetic computation units in a precision-on-demand configuration in order to reduce average power dissipation [13]. This method exploits the fact that not all spectral coefficients have the same visual significance in an image or video processing application. Typically, a large number of high spatial frequencies are quantized to zero in a lossy image/ video compression environment such as JPEG or MPEG with no significant change in visual quality. The DCT processor exploits such different precision requirements on a coefficient basis by reducing the number of iterations of the distributed arithmetic units that compute the visually insignificant spectral coefficients. A row-column classification scheme is implemented to further increase image quality while keeping arithmetic activity and power dissipation to a minimum: When the incoming pixel data exhibits increased (reduced) activity and low (high) correlation, the precision of the DA arith-

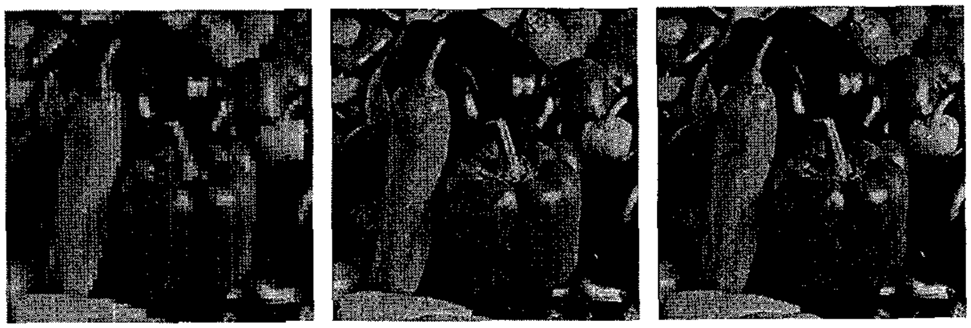

Figure 7: Compressed Image Quality and Power

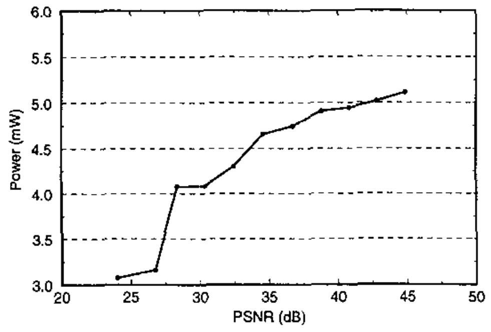

Figure 6: DCT Chip Average Power vs. Compressed Image Quality. Chip Measured Power at 1.56V, 14 MHz. The power measurements are for test image PEPPERS.

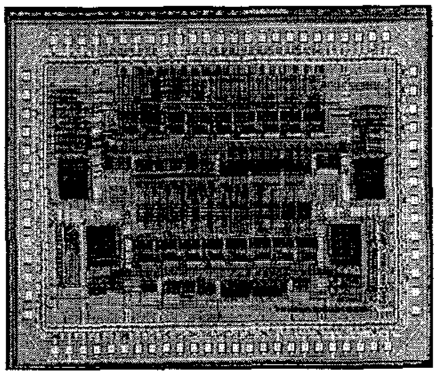

Figure 8: DCT Chip Microphotograph

metic units is dynamically increased (reduced).

The DCT core processor implements a row-column version of the Chen DCT algorithm [14] and operates on 8x8 blocks of pixel data. First the 8-point 1D DCT of each block row is computed, then the intermediate result is transposed and the same computation is performed on each block column. Each 1D computation stage can be described with the following two matrix equations:

$$\begin{bmatrix} X_0 \\ X_2 \\ X_4 \\ X_6 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} A & A & A & A \\ B & C & -C & -B \\ A & -A & -A & A \\ C & -B & B & -C \end{bmatrix} \cdot \begin{bmatrix} x_0+x_7 \\ x_1+x_6 \\ x_2+x_5 \\ x_3+x_4 \end{bmatrix}$$

$$\begin{bmatrix} X_1 \\ X_3 \\ X_5 \\ X_7 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} D & E & F & G \\ E & -G & -D & -F \\ F & -D & G & E \\ G & -F & E & -D \end{bmatrix} \cdot \begin{bmatrix} x_0-x_7 \\ x_1-x_6 \\ x_2-x_5 \\ x_3-x_4 \end{bmatrix}$$

where

$$A = \cos\frac{\pi}{4},\ B = \cos\frac{\pi}{8},\ C = \sin\frac{\pi}{8},\ D = \cos\frac{\pi}{16}$$

$$E = \cos\frac{3\pi}{16},\ F = \sin\frac{3\pi}{16},\ G = \sin\frac{\pi}{16}$$

Each vector inner product between a matrix row and the right-hand-side column vector of the above matrix equations is performed within a Distributed Arithmetic RAC very similar to the one shown in Figure 1. The number of RAC iterations are different depending on the spectral coefficient visual significance. The maximum number of iterations per RAC unit per input data class are programmed at reset time through an IEEE JTAG serial interface.

The chip average power dissipation varies with arithmetic precision as expected. Figure 6 plots average chip power dissipation vs. compressed image quality in terms of the image peak SNR (PSNR), a widely used quality measure in the image processing literature. The datapoints on the graph have been obtained by chip power measurements at different RAC maximum iteration settings. The measurements imply that the chip can produce on average 10 additional dBs of image quality per milliwatt of power dissipation.

Figure 7 displays the actual compressed images for three (power, PSNR) datapoints of Figure 6. Figures 6 and 7 establish our claim that the present chip trades-off image quality and power dissipation. A chip microphotograph is shown in Figure 8.

6 Conclusion

Approximate signal processing methods can be exploited for power scalable applications by trading off the quality of outputs for power consumption of the computation engine. In this paper, we have demonstrated that Distributed Arithmetic can be used in power scalable applications by varying the number of times the bit-serial units are clocked. Two DSP case studies, a DSP for physiological monitoring and a DCT processor, use this technique to achieve power scalability in addition to low power consumption.

Disclaimer

The views and conclusions contained in this document are those of the authors and should not be interpreted as presenting the official policies, either expressed or implied, of the Army Research Laboratory or the US Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

References

[1] J. T. Ludwig, S. H. Nawab, and A. P. Chandrakasan, "Low-power digital filtering using approximate processing," *IEEE JSSC*, vol. 31, no. 3, pp. 395–9, March 1996.

[2] C. Nicol, P. Larsson, K. Azadet, and J. O'Neil, "A low-power 128-tap digital adaptive equalizer for broadband modems," *IEEE JSSC*, vol. 32, no. 11, pp. 1777–1789, November 1997.

[3] P. Larsson and C. Nicol, "Self-adjusting bit-precision for low-power digital filters," in *1997 Symposium on VLSI Circuits Digest of Technical Papers*, June 1997, pp. 123–4.

[4] A. Peled and B. Liu, "A new hardware realization of digital filters," *IEEE Trans. ASSP*, vol. ASSP-22, no. 6, pp. 456–62, December 1974.

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

[6] T. Xanthopoulos, *Low Power Data Dependent Transform Video and Still Image Coding*, Ph.D. thesis, Massachusetts Institute of Technology, February 1999.

[7] S. Uramoto, Y. Inoue, A. Takabatake, J. Takeda, Y. Yamashita, H. Terane, and M. Yoshimoto, "A 100 MHz 2-D discrete cosine transform core processor," *IEEE Journal of Solid State Circuits*, vol. 36, no. 4, Apr. 1992.

[8] M. T. Sun, T. C. Chen, and A. M. Gottlieb, "VLSI implementation of a 16×16 discrete cosine transform," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 4, Apr. 1989.

[9] P.E. Landman and J.M Rabaey, "Architectural power analysis: The dual bit type method," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 3, no. 2, pp. 173–187, June 1995.

[10] C.-Y. Tsui, K.-K. Chan, Q. Wu, C.-S. Ding, and M. Pedram, "A power estimation framework for designing low power portable video applications," in *Proceedings of the 34th Design Automation Conference (DAC '97)*, June 1997, pp. 415–420.

[11] M. Nadler and E. Smith, *Pattern Recognition Engineering*, John Wiley and Sons, Inc., New York, 1993.

[12] T. Xanthopoulos and A. Chandrakasan, "A low-power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization," in *1999 Symposium on VLSI Circuits*, June 1998.

[13] S. H. Nawab, A. V. Oppenheim, A. P. Chandrakasan, J. M. Winograd, and J. T. Ludwig, "Approximate signal processing," *Journal of VLSI Signal Processing*, vol. 15, no. 1-2, pp. 177–200, January 1997.

[14] W. H. Chen, C. H. Smith, and S.C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Transactions on Communications*, vol. COM-25, no. 9, Sept. 1977.