Data Driven Signal Processing: An Approach for Energy Efficient Computing

Anantha Chandrakasan, Vadim Gutnik, Thucydides Xanthopoulos
Department of EECS,
Massachusetts Institute of Technology, Cambridge

ABSTRACT

The computational switching activity of digital CMOS circuits can be dynamically minimized by designing algorithms that exploit signal statistics. This results in processors that have time-varying power requirements and perform computation on demand. An approach is presented to minimize the energy dissipation per data sample in variable-load DSP systems by adaptively minimizing the power supply voltage for each sample using a variable switching speed processor. In general, using buffering and filtering, the computation can be spread over multiple samples averaging the workload and lowering energy further. It is also shown that four levels of voltage quantization combined with dithering is sufficient to closely emulate arbitrary voltage levels.

1. Introduction

The choice of algorithm to implement a given DSP function often has the greatest impact on reducing the number of switching events. To first order, the average power required to perform a signal processing task is:

$$P = \left(\sum N_i C_i V_{DD}^2\right) f_{sample} \quad (1)$$

where C_i is the average capacitance switched per operation of type i (corresponding to addition, multiplication, etc.), N_i is the number of operations of type i performed per sample, V_{DD} is the operating supply voltage, and f_{sample} is the sample frequency (which is fixed in many signal processing and dedicated applications).

An important goal of algorithm design for low-power implementations is to minimize the number of switching events, N_i. Typically, N_i is statically minimized at design time and is fixed per data sample. However, there are many systems where the amount of processing per input sample may naturally vary depending on data locality and correlations. For example, the number of times an MPEG decoder implements an inverse discrete cosine transform (IDCT) per frame varies widely depending on the temporal correlation between frames (Figure 1). More generally, as described in this paper, it is possible to design DSP algorithms exploiting signal statistics to dynamically minimize N_i, resulting in processors that have time-varying computational complexity.

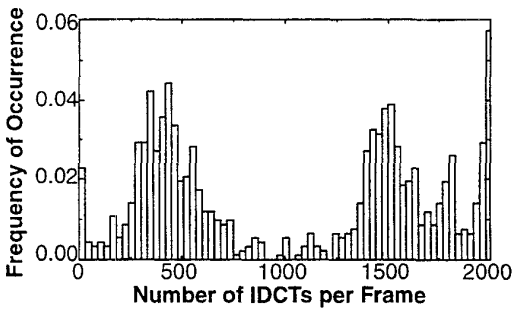

Figure 1. Typical IDCT histogram for MPEG video decoding

While the amount of computation per sample (henceforth called *computational workload* or simply *workload*) can vary drastically, the amount of available time to process a sample is fixed in throughput constrained systems. In the above example, an MPEG decoder must decompress a full frame of video every 30 msec but the amount of work per frame varies significantly. A generic approach is presented to lower the energy dissipation per input sample by increasing the switching time per computation and dynamically lowering V_{DD} during reduced processing loads instead of working at a fixed high voltage and allowing the processor to idle.

2. Optimizing Fixed Workload Systems

For DSP circuits that are continuously switching at a fixed rate (i.e., constant workload), the most efficient approach to lower energy consumption is to operate at the lowest possible power supply voltage. The individual circuit elements, however, run slower at lower supply voltages and circuit performance degrades. One approach to maintain throughput at reduced voltages is to use parallel architectures to compensate for increased gate delays at low voltages [1]. This approach, however, comes at the cost of increased silicon area.

Reducing the threshold voltage (V_T) of the devices also allows the supply voltage to be scaled down without loss in performance. Figure 2 shows a plot of energy vs. V_T for a two different speeds of a 101 stage ring oscillator (i.e., all points on each curve have a fixed delay). Here, the power supply voltage is allowed to vary to keep the performance fixed. These experimental plots are obtained from ring oscillator structures by adjusting the V_T (using the variable V_T dual-gated SOI device described in [2]) and V_{DD} for a fixed delay.

ISLPED 1996 Monterey CA USA
0-7803-3571-8/96/$5.00©1996

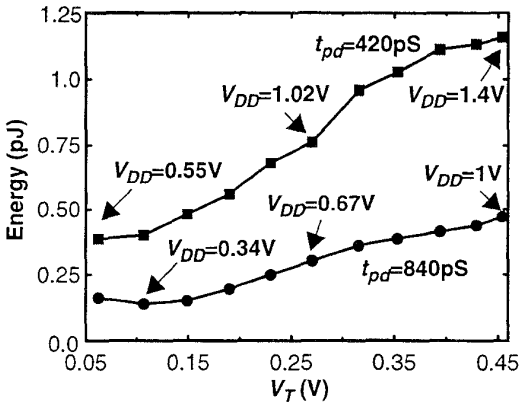

Figure 2. Experimentally derived optimum V_{DD}/V_T point using the dual-gate SOI technology described in [2].

For a fixed delay, the supply voltage and therefore the switching component of power can be reduced while reducing the threshold voltage. However, at some point, the threshold voltage and supply reduction is offset by an increase in the leakage current, resulting in an optimum threshold voltage and power supply voltage. That is, the optimum threshold voltage must compromise between lower switching energy and higher sub-threshold leakage energy at low supply voltages. It is interesting to note that the optimum supply voltage is significantly lower than 1V, even with a very low activity factor.

3. Variable Workload Algorithms

In some applications, like MPEG-2, the workload naturally varies with time. It is also possible to design or restructure algorithms to exploit data dependencies and adaptively minimize the number of switching events. When the computation performed is less than the peak, the processor can be shut down (and in general slowed down) to lower power. This results in processors that have variable workload. Two examples of such algorithm restructuring are presented below.

3.1 A Data Driven IDCT Algorithm

MPEG based video compression algorithms use the Discrete Cosine Transform (DCT) to remove correlation between pixel values. Analysis of MPEG video streams indicates that compressed video data contains a large percentage of zero-valued spatial frequency coefficients. The occurrence of zero-valued coefficients is the primary reason for the use of transform coding in the MPEG compression standard. Figure 3 shows the histogram of the relative occurrence of non-zero coefficients for a typical MPEG-2 sequence. The average number of non-zero coefficients for this sequence is 6.37 per 8x8 block.

Conventional 2D IDCT approaches are based on row column decomposition: first the 8-point 1D IDCT of each row is computed, then the result is transposed and the 8-point 1D IDCT of each column is computed. These

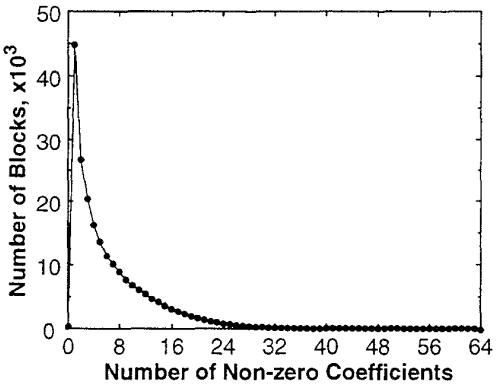

Figure 3. Histogram of non-zero DCT coefficients in a sample MPEG-2 stream.

approaches do not exploit data distribution to reduce energy dissipation. They perform a fixed number of operations per block independent of the data profile.

An interesting direct realization approach has been proposed for the implementation of the 2D 8x8 IDCT [3]. The forward mapped IDCT Algorithm (FMIDCT) can be formulated as follows:

$$\hat{x} = y_0\vec{C}_0 + y_1\vec{C}_1 + y_2\vec{C}_2 + \ldots + y_{63}\vec{C}_{63} \quad (2)$$

where $\hat{x}$ is the 64-element reconstructed image block, y_0, y_1, ..., y_{63} are the DCT coefficients, and $\vec{C}_0$, $\vec{C}_1$, $\vec{C}_2$, ..., $\vec{C}_{63}$ are 64-element constant reconstruction vectors.

As seen from Equation 2, each DCT coefficient is treated individually. Using this formulation combined with the observation that multiplication with a zero is a NOP, we can adaptively minimize the number of switching events by processing only the non-zero coefficients. This results in a variable number of operations per data sample (in this case an 8x8 block).

Figure 4 shows a plot of the number of multiplications per 8x8 compressed block for the data driven IDCT algorithm as well as a typical conventional algorithm (Chen's

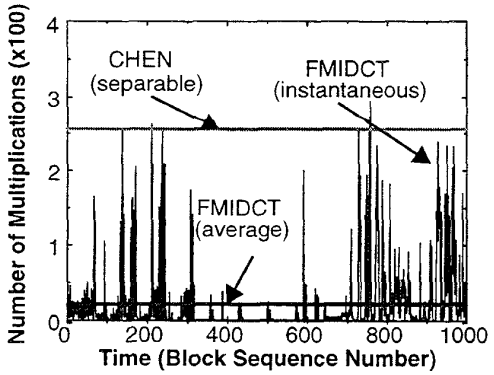

Figure 4. Multiply workload for DDIDCT vs. conventional.

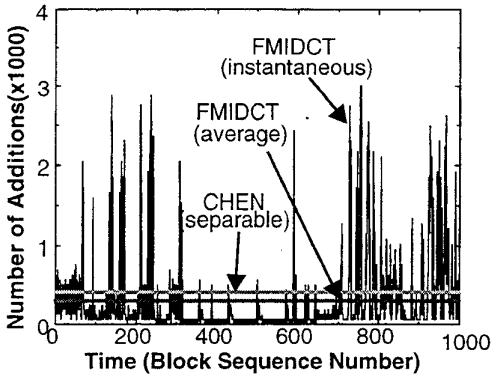

Figure 5. Addition workload for DDIDCT vs. conventional.

algorithm) implemented using a row-column arithmetic approach. While the instantaneous number of operations can be larger in the data driven approach, the average number of operations is significantly smaller. Figure 5 shows the number of additions for the sample stream.

3.2 Approximate Filtering Techniques

In many contexts it is important to structure algorithms and systems to allow for the possibility of trading off between the accuracy or optimality of the result and the utilization of resources such as power, time, bandwidth, etc. The formal approach to trade-off quality for resources has been termed "approximate signal processing" [4]. A well established example in communications involves the use of scalable video coding algorithms to trade quality for bitrate as the available channel bandwidth varies. Scalable architectures are required to minimize energy consumption of processors when the quality requirements on computational results change.

Approximate signal processing concepts can be applied to the design of adaptive digital filters [5]. Adaptive filtering algorithms have generally been used to dynamically change the values of the filter coefficients, while maintaining a fixed filter order. In contrast, the approximate processing approach involves the dynamic adjustment of the filter order. The basic idea is to exploit the fact that in many frequency-selective filter applications, the out of band signal ("noise") can vary with time and therefore the filter order can be dynamically adjusted. The idea is to keep the stopband energy in the filter output below a specified threshold while using as small a filter order as possible.

Figure 6 shows a conceptual block diagram of a variable length tapped delay line. Since power consumption is proportional to filter order, this approach achieves power reduction with respect to a fixed-order filter whose output is similarly guaranteed to have stopband energy below the specified threshold. Power reduction is achieved by dynamically minimizing the order of the digital filter.

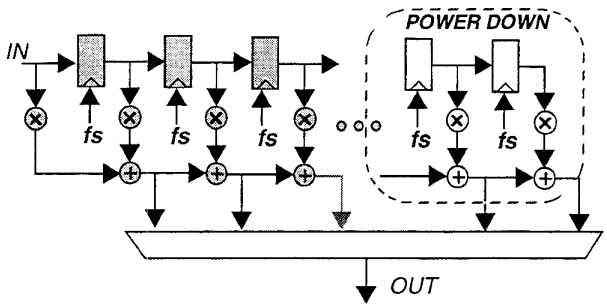

Figure 6. Conceptual diagram of a variable length tapped delay line.

The key challenge in such a filtering system is to determine the filter order as a function of time. Let a signal, $x[n]$, consist of a passband component, $x_p[n]$, and a stopband component, $x_s[n]$ (i.e., $x[n] = x_p[n] + x_s[n]$). If it were possible to *cost-effectively* measure the strength of the stopband component, $x_s[n]$, from observation of $x[n]$, the amount of stopband attenuation needed at any particular time can be determined. When the energy in $x_s[n]$ increases, it is desirable to increase the stopband attenuation of the filter. This can be accomplished by using a higher-order filter. Conversely, the filter order may be lowered when the energy in $x_s[n]$ decreases. A practical low-overhead approach is presented in [5], using feedback techniques, for dynamically estimating the energy fluctuations in the stopband component, $x_s[n]$, and using them to adjust the order of a frequency-selective filter.

The amount of power reduction is a strong function of signal statistics. Figure 7 illustrates the nature of the adaptation performed for a frequency division multiplexing example where the out of band signal varies [5]. One of the curves shows the evolution of the filter order while the other curve shows the energy profile of the stopband signal. Clearly, the variations in filter order roughly follow the energy variations of the stopband signal. In particular, the most power savings is achieved during the silent regions of the stopband signal.

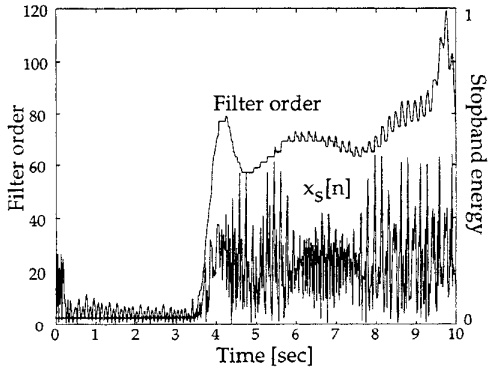

Figure 7. Filter order vs. time [5].

4. Variable Voltage Digital Signal Processing

From the previous section, it is clear that there are many applications in DSP where the computational workload varies with time. An approach to reduce the energy consumption of such systems beyond shut down involves the dynamic adjustment of supply voltage based on computational workload. The basic idea is to lower power supply voltage when the workload is less than peak instead of working at a fixed supply and idling for some fraction of the time. Self-timed circuits have been suggested as a means to exploit low-level data dependent variable computation times to vary power supply voltage, but the switched capacitance overhead due to dual-rail coding can be substantial [6].

We propose to use standard *synchronous* design with a ring-oscillator based clock and simple switching supply (Figure 8) [7]. The basic idea is to exploit time-varying computational complexity at an *algorithmic* level to vary the supply voltage. The supply voltage and clock rate are increased during high workload periods and reduced during low workload periods. Even if workload is fixed, energy can be lowered by tracking the process and temperature variations, thus keeping voltage at the lowest level possible [8],[9].

The supply voltage is adjusted by changing the duty cycle of the switching supply based on the workload derived from the input signal. The clock frequency is controlled by a digital phase-locked loop with the variable V_{DD} as the control signal for the ring-oscillator. If the ring oscillator is well matched to the circuits, logic should function properly, since logic transitions take place in nanoseconds, while voltage changes in microseconds for typical switching supplies. A lookup table (in the rate controller) records the voltage needed for each processing rate, and the feedback slowly adjusts the entries in the table as needed. Thus the loop tracks process and temperature, but computation rate is changed open-loop by reading different table entries. This PLL is similar to the ones presented in [8],[9] with appropriate modifications required to allow fast changes in supply voltage.

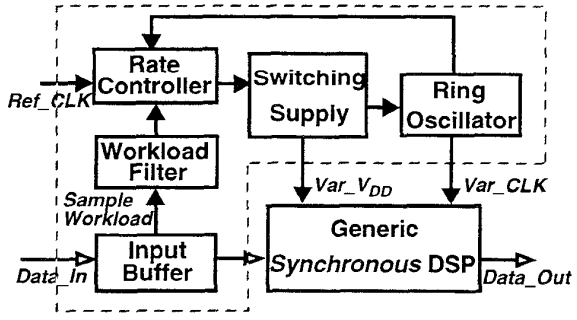

Figure 8. Variable supply voltage block diagram [7].

4.1 Energy Reduction Using Variable Supply

Circuits with a fixed supply voltage work at a fixed speed and idle if the data sample requires less than the maximum amount of computation. Less power would be consumed if voltage is lowered when less work needs to be done. Figure 9 shows a plot of power vs. normalized workload ($V_{DD(max)}$=2V and V_T=0.4). The straight line represents a fixed power supply system. If the workload is less than peak workload (i.e., workload =1), then the processor would compute as fast as possible at a fixed voltage and idle for (1-workload) fraction of sample period. Power is reduced in a linear fashion since the energy per operation is fixed, and only the number of operations changes.

The lower curve represents the case for variable power supply system. If the workload for a given sample period is less than peak, then the delay of the processing element can be increased by a factor of 1/workload without loss in throughput, allowing the processor to operate at a lower power supply voltage. The energy per sample therefore varies not only because fewer operations are performed, but also because each operation consumes lower energy (since the circuits are switching at a lower supply voltage). The difference between the two curves represents the energy savings.

4.2 Improving Energy Efficiency Using Averaging

We will make the assumption that it is possible to determine how much processing (i.e., the workload) will be needed for each block before any processing is done. In fact, in many cases this is valid - for example, MPEG protocols can specify how many blocks are to be decoded with each frame. The performance of such a system bounds the performance of systems where this information is not available, (e.g. error-correcting decoders [6]) and the problem turns out to be conveniently tractable.

In the previous section, it was shown that minimizing the voltage for each data sample can lower energy over simple shut down. If the latency of the system is allowed to increase, it turns out that averaging the workload over

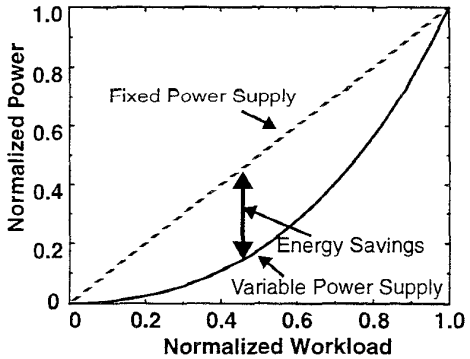

Figure 9. Power reduction using variable supply.

multiple samples lowers power further [7]. The basic idea of averaging two samples is shown in Figure 10. Two data samples are buffered (I_n and I_{n+1}) and their workloads (x_n and x_{n+1}) are averaged. The averaged workload $(x_n + x_{n+1})/2$ is then used as the effective workload to drive the power supply. Using a ping-pong buffering scheme, data samples I_{n+2} and I_{n+3} are being buffered while I_n and I_{n+1} are being processed. Note that the averaging approach does not guarantee that each sample is finished during its sample period, but does guarantee that two output samples will be ready after two sample periods.

Figure 11 shows the graphical interpretation of power reduction using the averaging technique. Assume that the sample workload alternates between 0.2 and 0.8. For the fixed power supply case, the average power to process two samples will be $(E_{workload=0.2} + E_{workload=0.8})/2 = (0.2 + 0.8)/2 = 0.5$. This point lies on the top line. Next consider the case when the voltage is optimized individually for each sample. The average energy per sample is the average of the energy corresponding to points (a) and (b). This average lies on the line connecting points (a) and (b). Finally consider the averaging approach. The basic idea is to buffer two samples and average the workload. The effective workload is (0.2 + 0.8)/2 = 0.5. This averaged workload is used to drive the power supply and results in the point on the lower curve. This simple example demonstrates that averaging can lower power beyond varying the power supply on a sample by sample basis.

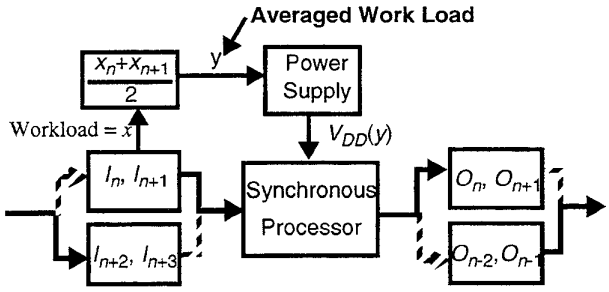

Figure 10. Schematic representation of averaging.

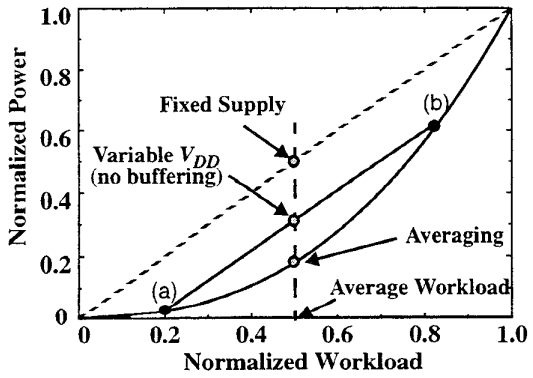

Figure 11. Power reduction using averaging.

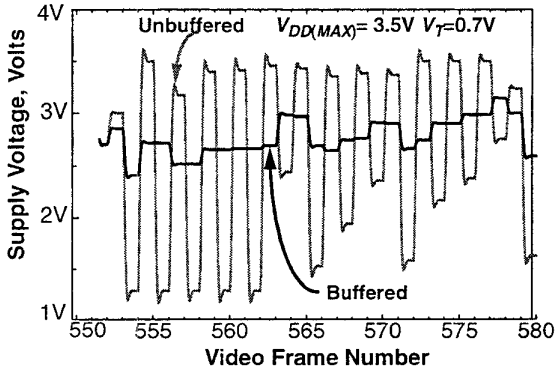

Figure 12. Effect of averaging workload on V_{DD}.

In general, it can also be shown that averaging computational load over several samples saves power, but any low-pass FIR filter can be used to smooth x[n] (workload filter shown in Figure 8). Note that blocks of data are not finished at discrete times, so a separate register is needed to keep track of x[n]. Formally,

$$y[n] = \sum_{i=1}^{L} a_i x[n-i] \qquad (3)$$

where $\sum_{i=1}^{L} a_i = 1 \; ; \; (a_i > 0)$.

Figure 12 shows voltage requirements for a variable-voltage MPEG decoder with and without buffering. Although buffering lowers the power, and there are some cases where buffering is necessary (e.g., variable-rate packet arrival over a network), buffering is not inherent in this synchronous scheme the way it is in the self-timed approach.

4.3 Voltage Quantization

An important issue is the number of voltage levels required for efficient variable voltage processing. Figure 13 shows a plot of power *vs.* workload for four cases: fixed supply, infinite levels, and two plots for a four-level system [7]. For simplicity, assume that workload is fixed for some time interval. The bottom curve assume that there are infinite voltage levels and therefore the processor delay can always be stretched out to exactly to fill up the sample period - i.e., the processor never idles. The step curve is for four levels of voltage quantization and assuming that only one block is processed per sample period (i.e., no averaging of workload). In this case, the processor will idle for part of the sample period unless the computation workload exactly matches the quantized voltage level.

If the processing of one data block can cross sample-time boundaries (i.e., the data is buffered) the processor will never idle: whenever the actual workload happens to

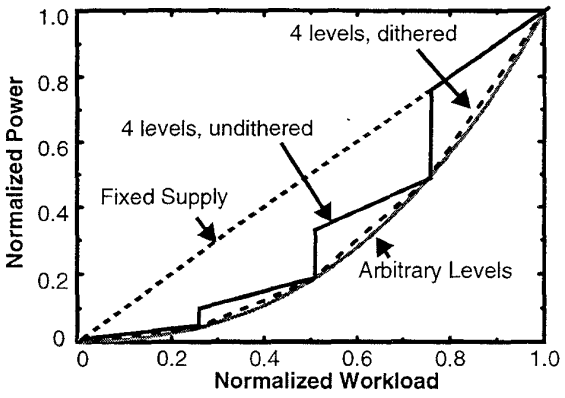

Figure 13. Quantization of voltage levels.

be between two fixed levels, some blocks will be processed at the higher rate and some at the lower, and the average power is given by the weighted sum of the two levels - the dotted curve just above the infinite levels. It is clear that even rough quantization (4-levels) is sufficient to get power savings close to the ideal case.

4.4 Relation to Parallelism

It has been shown that parallelism can be used to scale supply voltages without performance loss [1]. One approach to energy efficient computing is to operate at a fixed "optimum" supply voltage level dictated by parallelism and simply power down units when the workload is less than the peak. Figure 14 shows the energy consumption for various levels of parallelization with and without variable supply voltages (assuming no overhead circuitry). The top set of graphs correspond to a reference system with some arbitrary level of parallelism. The middle set of graphs correspond to a parallelization by a factor of 2 with respect to the reference system. With a total peak workload of 1, each unit can operate at a workload of 0.5 and therefore can operate a lower voltage compared to the reference system. A variable voltage scheme can be used with the parallel system if the workload is less than the peak. The energy savings relative to simply powering down the unit at a fixed voltage is somewhat

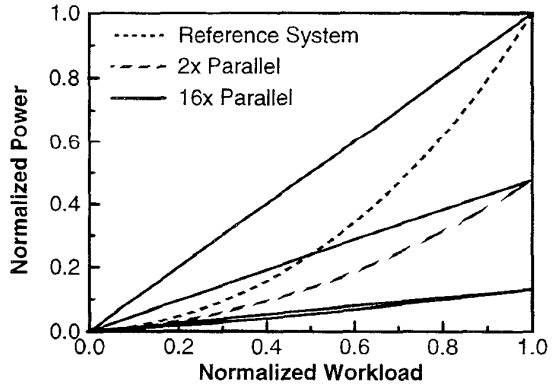

Figure 14. Comparison of variable voltage with parallelism.

smaller than the reference system. The figure also shows the case for 16X parallelism. As evident from the figure, there is little to gain by using a variable voltage there.

The figure also indicates that if the typical workload is small, then the variable supply scheme can approach the energy efficiency of the parallel scheme. This is attractive since the implementation area can be significantly reduced while achieving significant power reduction.

5. Conclusion

Time varying computational requirements of many signal processing algorithms can be exploited to reduce power dissipation by using a variable power supply voltage. Buffering data and averaging workload allows a trade-off between latency and power dissipation. A few voltage levels combined with dithering are sufficient to achieve the effect of infinite levels. The key message is to achieve low-power in variable-load DSP systems, the *power supply should be embedded into the computation!*

Acknowledgments

This work is sponsored in parts by ARPA contract # DABT63-95-C-0088, NSF Career Development Award MIP-9501995, and SHARP Corporation. Vadim Gutnik is supported by an NDSEG fellowship. The authors thank I. Yang and Prof. D. Antoniadis for the experimental data from the dual-gated SOI technology.

References

[1] A. Chandrakasan, R. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, July 1995.

[2] I. Yang, C. Vieri, A. Chandrakasan, and D. Antoniadis, "Back Gated CMOS on SOIAS for Dynamic Threshold Control," *IEEE 1995 IEDM*, December 1995.

[3] L. McMillan, L. Westover, "A Forward-Mapping Realization of the Inverse Discrete Cosine Transform," *Data Compression Conference*, pp. 219-228, March 1992.

[4] A. Oppenheim, H. Nawab, G. Verghese, and G. Wornell, "Algorithms for Signal Processing," *1st RASSP Conference*, pp. 146-153, 1994.

[5] J. Ludwig, H. Nawab, A. Chandraksan, "Low-Power Digital Filtering Using Approximate Processing," *IEEE Journal of Solid-State Circuits*, pp. 395-400, March 1996.

[6] L. Nielsen, C. Niessen, J. Sparso, K. van Berkel, "Low-Power Operation Using Self-Timed Circuits and Adaptive Scaling of Supply Voltage," *IEEE Transactions on VLSI systems*, pp 391-397, December 1994.

[7] V. Gutnik, A. Chandrakasan, "An Efficient Controller for Variable Supply-voltage Low Power Processing," *IEEE VLSI Circuits Symposium*, June 1996.

[8] P. Maken, M. Degrauwe, M. Van Paemel, H. Oguey, "A Voltage Reduction Technique for Digital Systems," *IEEE ISSCC*, pp. 238-239, February 1990.

[9] M. Horowitz, "Low Power Processor Design Using Self-Clocking," *Workshop on Low-power Electronics*, 1993.