Ultra Low Power Digital Signal Processing

Anantha P. Chandrakasan
Massachusetts Institute of Technology, Cambridge, MA

ABSTRACT

The explosive growth of portable wireless devices has elevated power consumption to be one of the most critical design parameters. This paper presents several techniques to implement DSP functions with the lowest possible power consumption. Since power is consumed only when capacitance is being switched, power can be reduced by minimizing this capacitance through operation reduction, choice of data representation, exploitation of spatial and temporal signal correlations, re-synchronization to minimize glitching, and logic, circuit and physical design. Aggressive voltage scaling to 1V and below through circuit and architecture optimization is the key to low-power design. A systems approach to low-power design can result in orders of magnitude power reduction.

1. Introduction

The weight and size of batteries, which are major factors in portable systems, are primarily determined by the power dissipation of electronic circuits. Until recently, there has not been a major focus on a design methodology of digital circuits which directly addresses power reduction, with the focus rather on ever faster clock rates and logic speeds. The emphasis of this work is on minimizing the power consumption in DSP applications, which have two important attributes that make then *different* from general purpose computation: throughput constrained computing and correlation in the data being processed.

The throughput constraint implies that once the required sample rate is met by the hardware, there is no advantage in making the computation faster (unlike general purpose computation where is goal is to compute as fast as possible). This enables a variety of voltage scaling techniques that lower power while maintaining a constant throughput [1]. The second attribute of signal processing systems that can be exploited for low-power is the correlation present in the data being processed. Unlike general purpose computation, the data being processed in many signal processing applications can exhibit high temporal correlation. An important goal of low-power architecture selection is to preserve naturally occurring signal correlations and minimize switching activity.

The focus of this paper is on CMOS circuits and it is well known that power is primarily dissipated in charging and discharging parasitic capacitors and is given by $\alpha_{0->1} \bullet C_L \bullet V_{dd}^2 \bullet f_{clk}$, where C_L is the load capacitance, V_{dd} is the power supply voltage, f_{clk} is the clock frequency, and $\alpha_{0->1}$ is the node transition activity factor (or the fraction of time the node makes a power consuming transition inside a clock period). For example, Figure 1 shows the activity factor, $\alpha_{0->1}$, for a simple 2-input NOR gate as a function of the input signal probabilities, p_a and p_b.

$$\alpha_{0->1} = p_0\, p_1 = (1-(1-p_a)\,(1-p_b))\,(1-p_a)\,(1-p_b) \quad \text{(EQ 1)}$$

The strong influence of signal statistics on power implies that the data statistics must be carefully considered during power optimization. Signal statistics should be considered at all levels of the design including the selection of data representation, resource allocation, operation sequencing, logic structuring, and place and route.

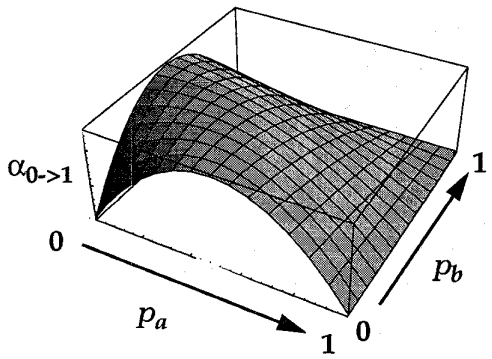

Figure 1: Dependence of power on signal statistics.

2. Optimizing Data Representation

Two's complement is typically chosen to represent numbers in signal processing applications since arithmetic operations are easy to perform. One of the problems with two's complement representation is sign-extension, which causes the msb sign-bits to toggle when a signal changes sign (for example, going from

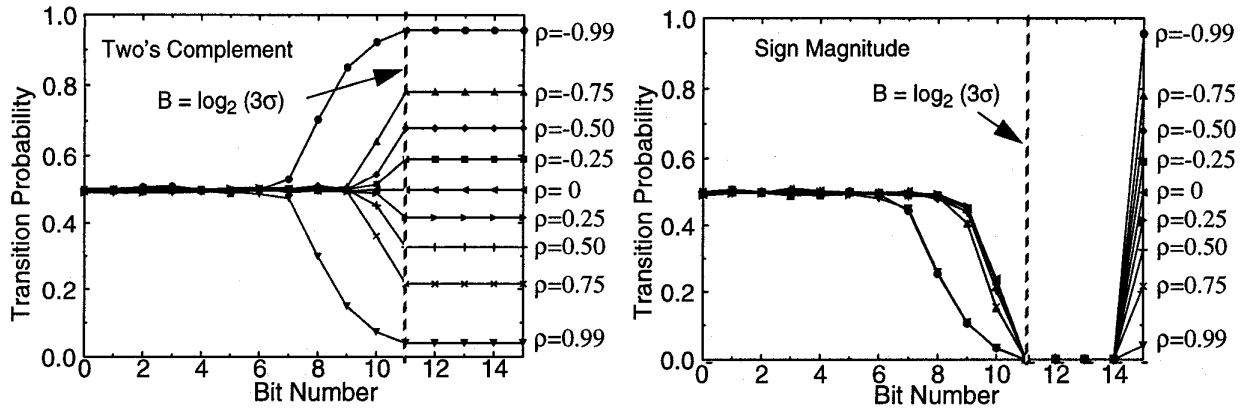

Figure 2: Reducing switching activity by optimizing number representation.

-1 to 0 will result in all of the bits toggling). This is especially significant when the dynamic range of the signals being processed is much smaller than the peak value for the bit-width used since a lot of the msb bits will perform sign-extension. Even when a signal spans the entire bit-width, scaling operations can reduce the dynamic range.

An approach to minimizing the switching activity in the msbs is to use a sign-magnitude representation, in which only one bit is allocated for the sign and the rest to the magnitude [1]. In this case, only one of the msb bits will toggle when the signal switches sign, as opposed to the two's complement representation where due to sign-extension several of the bits will switch. To illustrate this, consider Gaussian data applied to a 16-bit data-bus, which has a mean of 0 and a $3\sigma = 2^{11}$ (the upper breakpoint which represents the dynamic range of the signal lies at $\log_2(3\sigma) = 11$ bits in this case). Figure 2a shows the transition probabilities (both the 0->1 and 1->0) vs. bit position number for two's complement representation as a function of the first order correlation coefficient. A ρ=0 indicates that the data is uncorrelated and therefore the transition probability for the msb's is 1/2 (i.e., there is a 50% chance the output is going to transition between positive and negative). A large positive correlation coefficient implies that the signal changes very slowly and therefore switches sign very infrequently. Similarly, a negative correlation coefficient with a large magnitude implies that the signal changes sign frequently. In two's complement representation, bits 11 to 15 are used for sign-extension. If a sign-magnitude representation is used (Figure 2b), the higher order bits will have a low transition activity and bit 15 (the sign bit) will have the same activity as the two's complement case.

The reduction in the number of transitions for sign-magnitude over two's complement is a function of both the dynamic range of the signal and the signal correlation coefficient. Figure 3 shows the ratio of the number of transitions required in the two's complement representation to that required in the sign-magnitude representation as a function of the signal correlation and normalized dynamic range. The normalized dynamic range of a signal is defined as $3 * \sigma$ / Peak Amplitude and the number of transitions is the sum of the transitions for all the bits per clock cycle (i.e., number of transitions = $\sum \alpha_i$, $0 \leq i \leq 15$). The biggest advantage for using sign-magnitude is when the dynamic range is small and the signal has a large negative correlation. Sign-magnitude is clearly useful for driving large busses, where the overhead for converting back to two's complement is quite insignificant compared to the reduction in capacitance switched in the large busses. The scheme presented here is only one example of coding data for reducing power. There are several other schemes which can be used such as log representation, gray-coding [2], conditional inversion coding [3], etc.

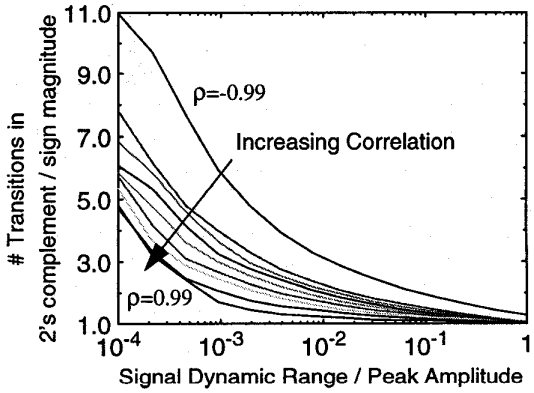

Figure 3: Activity reduction using sign-magnitude.

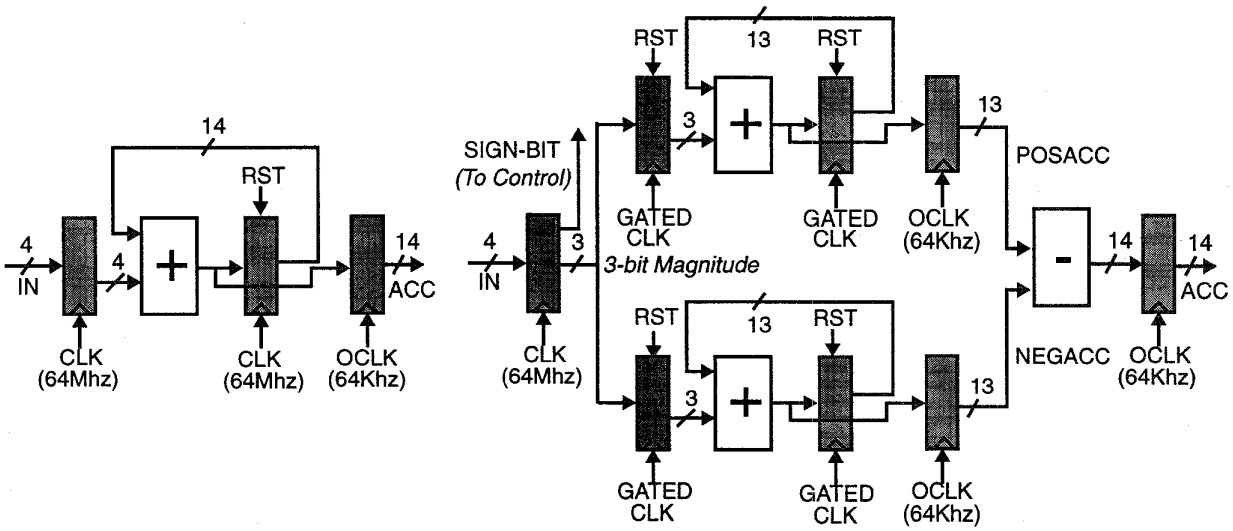

Figure 4: Number Representation trade-off in arithmetic computation: accumulator example.

While sign-magnitude has some advantages in terms of the number of transitions on busses, addition/ subtraction operations are difficult to implement. However, in several applications, that require multiple additions inside a sample period (e.g., Multiply Accumulate Units or DSP filters), an architecture optimization can be used that trades silicon area for lower switching activity without altering the throughput. To illustrate this consider a CDMA radio correlator example in which the correlation length is 1024; the samples, whose values ranges from -7 to +7, are accumulated at 64Mhz.

On the left side of Figure 4, is a conventional architecture for an accumulator that uses two's complement representation. The accumulated result is transferred to an accumulator register at 64Khz. In this architecture the msb-bit, bit 3, is tied to bits 4-13 of the adder input for sign-extension and therefore the adder has high switching activity in the msb-bits (due to glitching) when the input changes sign frequently.

Another approach is to use a sign-magnitude representation (right side of Figure 4). Here two accumulator datapaths are used, one that sums all positive numbers and one that sums all negative numbers. The latched sign-bit (from the input register) is used to generate gated clocks that enables the positive datapath latch or the negative datapath latch (this ensures that this scheme does not increase effective clock load). At the end of 1024 cycles, the positive and negative accumulated values are transferred to separate registers. A subtract operation is then performed at the low-frequency and therefore is quite negligible in terms of capacitance overhead. In fact, the higher order bits (3-13) only need an incrementer (as opposed to a full-adder in the two's complement case), reducing the number of gates that are switched in the accumulator. The sign-magnitude implementation, however, requires control circuitry (overhead capacitance) to generate the timing signals for the various latches in the implementation.

By keeping the computation for positive data and negative data separate, the power is not very sensitive to rapid fluctuations in the input data. Table 1 shows the power numbers for various input patterns. Again, the biggest advantage is when the sign toggles frequently. For the case when the input changes very slowly, the sign-magnitude implementation consumes more power (15%) due to the capacitance switched by the control overhead circuitry. For the radio example, data coming in to the accumulator tends to be random.

Table 1. Number representation trade-off for arithmetic.

Input Pattern (1024 cycles)	Two's Complement Power, 3V	Sign Magnitude Power, 3V
Constant (IN= 7)	1.97 mW	2.25 mW
Ramp (-7,-6,...7,-7)	2.13 mW	2.43 mW
Random	3.42 mW	2.51 mW
Min -> Max -> Min (-7, +7,-7,+7,...)	5.28 mW	2.46 mW

3. Time-sharing of Resources

One strategy for implementing signal processing algorithms is the direct mapping approach, where there is a one to one correspondence between the operations on the signal flow graph and operators in the final implementation. Such an architecture style is

conceptually simple and requires a small or no controller. Often, however, due to area constraints or if very high-throughput is not the goal (e.g., speech filtering), time-multiplexed architectures are utilized in which multiple operations on a signal flowgraph can be mapped onto the same functional hardware unit. Given that there is a choice between time-multiplexed and fully-parallel architectures, as is the case in low to medium throughput applications, an important question which arises is what architecture will result in the lowest switching activity. To first order, it would seem that the degree of time-multiplexing would not affect the capacitance switched by the logic elements or interconnect. For example, if a data flowgraph has five additions to be performed inside the sample period, it would seem that there is no difference between an implementation in which one physical adder performs all five additions or an implementation in which there are five adders each performing one addition per sample period. It would seem that the capacitance switched should only be proportional to the number of times the additions were performed but this is not the case since resource sharing can destroy signal correlations and increase switching activity. To demonstrate this trade-off, consider two examples of resource sharing: sharing busses and sharing execution units.

First consider two implementations of two 8-bit counters clocked every cycle one with bus sharing and one without. In the parallel implementation, the two counters are driving two separate busses running at frequency f while the time-multiplexed implementation has a single shared bus running at twice the frequency. For the parallel case, the activity is easy to compute. The number of transitions for each bus = $\Sigma\alpha$ = 1 + 1/2 + 1/4 + ... + 1 / 128 ≈ 2 since the lsb of the counter switches every cycle, the 2nd lsb switches every other cycle, etc. Therefore, the total number of transitions per cycle is approximately 4 for the parallel implementation.

Figure 5 shows the plot of the average number of transitions for the time-shared case as a function of the skew between the counter outputs. The figure also shows the number of transitions for the non-time-shared implementation, which is independent of the counter skew. As seen from this figure, except for one value of the counter skew (when the skew is = 0), the parallel implementation has the lower switching activity. This example illustrates that time-sharing can modify the signal characteristics and cause an increase in the switching activity.

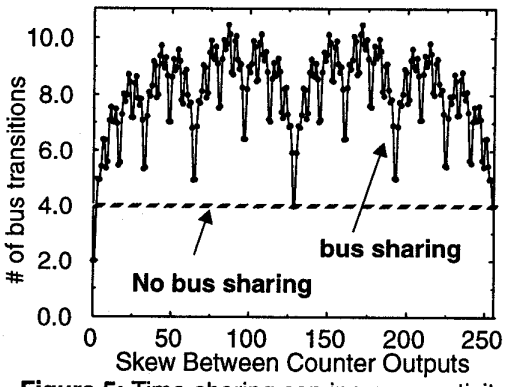

Figure 5: Time-sharing can increase activity.

The second example is a simple second order FIR filter which will demonstrate that time-multiplexing of execution units can increase the switched capacitance. The FIR filter is described as follows:

$$Y = a_0 \bullet X + a_1 \bullet X@1 + a_2 \bullet X@2 = A + B + C \quad \textbf{(EQ 2)}$$

where @ represents the delay operator and $A = a_0 \bullet X$, $B = a_1 \bullet X@1$ and $C = a_2 \bullet X@2$. Also let $O1 = A + B$. The value of the coefficients are: a_0 = 0.15625, a_1 = 0.015625, and a_2 = -0.046875.

One possible implementation might be to have two physical adders, one performing $A + B = O1$ and the other performing $O1 + C$. An alternate implementation might be to have a single time-multiplexed adder per-

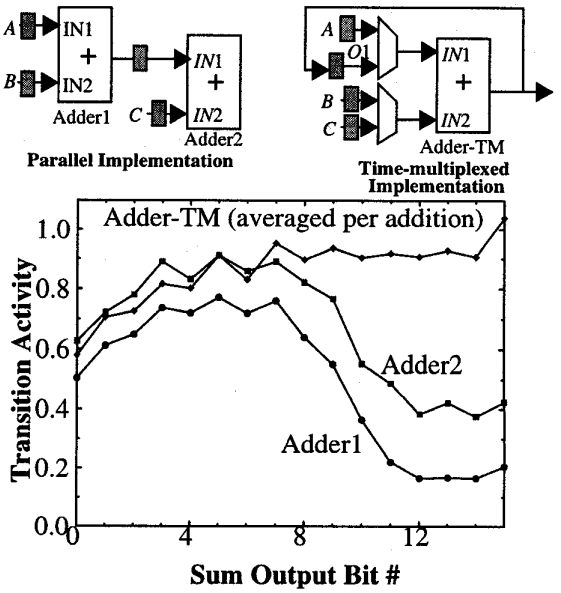

Figure 6: Activity trade-off for time-multiplexed hardware: adder example.

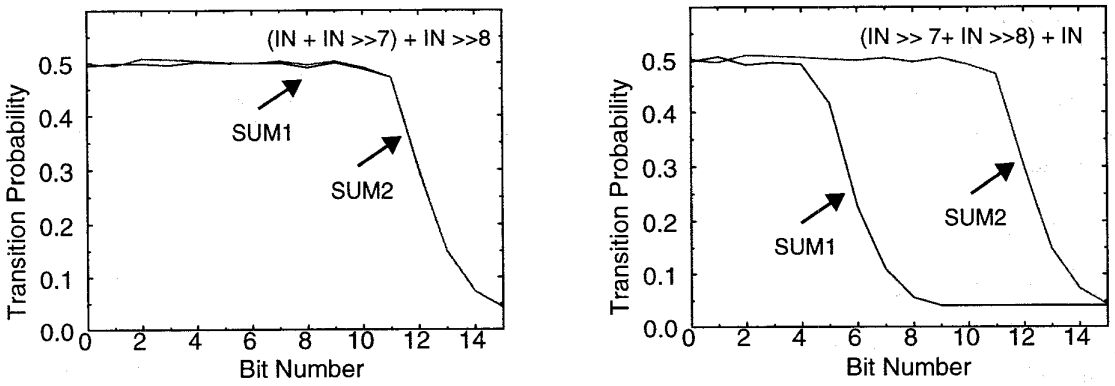

Figure 7: Reducing switching activity by re-ordering inputs.

forming both additions. So, in cycle 1, $A + B$ is performed, and in cycle 2, $O1 + C$ is performed. The topologies for the two cases are shown in Figure 6. In the first implementation, the adders can be chained and therefore the effective critical path can be reduced (chaining two ripple carry adders of N bits has a delay = $(N + 1)$ • Delay of one bit); it is clear that this will result in extra glitching activity but since the objective here is to isolate and illustrate the activity modification only due to time-multiplexing, the first implementation is assumed to be registered. The transition activities (obtained using IRSIM assuming speech input data) for the two adders for the parallel implementation and the average transition activity per addition for the time-multiplexed implementation are shown in Figure 6.

The results once again indicate that time-multiplexing can increase the overall switching activity. The basic idea is that in the fully-parallel implementation, the inputs to the adders change only once every sample period and the source of inputs to the adders are fixed (for example, *IN*1 of adder1 always comes from the output of the multiplier $a_0 * X$). Therefore, if the input changes very slowly (i.e., the input data is very correlated), the activity on the adders become very low. However, in the time multiplexed implementation, the inputs to the adder change twice during the sample period and more importantly arrive from different sources. For example, during the first cycle, *IN*2 of the time-multiplexed adder is set to B and during the second cycle, *IN*2 of the time-multiplexed adder is set to C. Thus, even if the input is constant (which implies that the sign of X, $X@1$ and $X@2$ are the same) or slowly varying, the sign of B and C will be different since the sign of the coefficients a_1 and a_2 are different. This will result in the input to adder switching more often and will therefore result in higher switching activity. That is, even if the input to the filter does not change, the adder can still be switching. For this example, the time multiplexed adder switched 50% extra capacitance per addition compared to the parallel case (even without including the input changes to adders or the multiplexor overhead).

4. Optimizing Ordering of Operation

The switching activity can be reduced by optimizing the ordering of operations in a design. To illustrate this, consider the problem of multiplying a signal with a constant coefficient and assume that the multiplication is decomposed into IN + IN >>7 + IN>>8. Figure 7 shows the results of two alternate implementations for the two required additions. In the first implementation, IN and IN >>7 are added first and the sum (SUM1) is then added to IN >>8. In this case, the SUM1 transition characteristics is very similar to that of the input IN since the amplitude of IN >>7 is much smaller than IN and the 2 inputs have identical sign-bits. Similarly SUM2 is very similar to SUM1 since SUM1 is much larger in magnitude than IN >>8. In the second implementation, the two small number IN>>7 and IN>>8 are added first and the output SUM1 is then added to IN. In this case, the output of the first adder has a small amplitude and therefore lower switching activity. The second implementation switches 30% less capacitance than the first implementation for this data.

5. Voltage Scaling

Since the dominant component of power consumption for a properly designed CMOS circuit is proportional to the square of the supply voltage, operating circuits at the lowest voltage is the key to minimizing the energy consumed per operation. However, the individual circuit elements run slower at lower supply voltages and this must be compensated for through appropriate architectural design. One important class of applications are those which have no advantage in exceeding a bounded computation rate, as found in

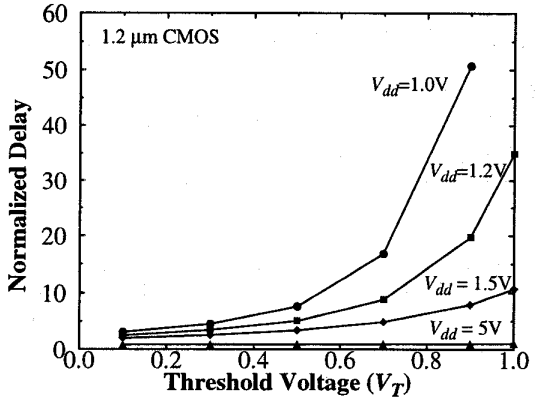

Figure 8: Effect of threshold reduction on the delay for various supply voltages.

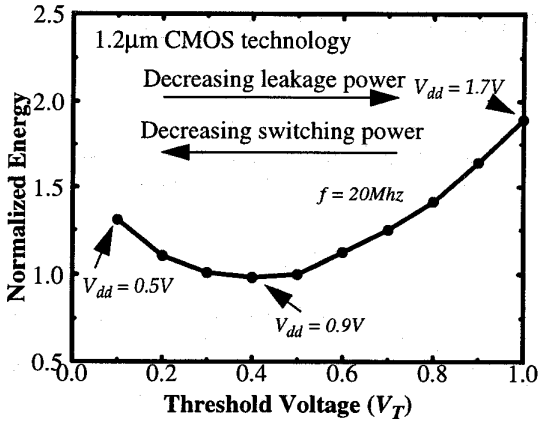

Figure 9: Compromise between dynamic and leakage power dissipation through V_t variation.

real-time signal processing. For such applications, we have previously presented an architecture driven voltage scaling approach in which parallel and pipelined architectures can be used to compensate for increased delays at reduced voltages [4][5]. As the amount of parallelism increases, the amount of overhead circuitry also increases and results in an optimum level of parallelism for low-power. The optimum voltage corresponding to this level of parallelism was found to be close to 1.5V ($\approx 2V_T$) using a standard technology with high threshold devices (V_{TN}=0.7V and V_{TP}=-0.9V).

Reducing the threshold voltage of the device allows the supply voltage to be scaled down (and therefore lower switching power) beyond the architectural optimum without loss in performance. For example, a circuit running at a supply voltage of 1.5V with V_T=1V will have approximately the same performance as the circuit running at a supply voltage of 0.9V and a V_T=0.5V. Figure 8 shows a plot of normalized delay vs. threshold voltage for various supply voltages.

Since significant power improvements can be gained through the use of low-threshold MOS devices, the question of how low the thresholds can be reduced must be addressed. Figure 9 shows a plot of energy vs. threshold voltage for a fixed throughput for a 16-bit datapath ripple carry adder (which essentially represents the power to perform the operation). Here, the power supply voltage is allowed to vary to keep the throughput fixed. For a fixed throughput (e.g., that is obtained at a 20Mhz clock rate), 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 currents, resulting in an optimal threshold voltage for a given level of logic complexity. That is, the optimum threshold voltage must compromise between improvement of current drive at low supply voltage operation and control of the sub-threshold leakage. Circuits have been designed with power supply voltages as low as 200mV using reduced threshold devices [6].

Acknowledgments

The work presented in this paper is a result of a collaboration with Prof. Brodersen and graduate students at U.C. Berkeley. This research was performed as a part of the InfoPad wireless terminal design.

References

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

[2] C. Su, C Tsui, and A. Despain, "Low-power Architecture Design and Compilation Techniques for High-Performance Processors," pp. 489-498, *Compcon 1994.*

[3] M. Stan and W. Burleson, "Limited-weight Codes for Low-power I/O," *1994 International Workshop on Low-power Design*, pp. 209-214, April 1994.

[4] A. Chandrakasan, S. Sheng, R. Brodersen, "Low-Power CMOS Digital Design", *IEEE Journal of Solid-State Circuits*, pp. 473-484, April 1992.

[5] A. Chandrakasan, A. Burstein, and R. Brodersen, "A Low-power Chipset for a Portable Multimedia I/O Terminal," *IEEE Journal of Solid-State Circuits*, December 1994.

[6] J. Burr, J. Shott, "A 200mV Self-Testing Encoder/Decoder using Stanford Ultra-low Power CMOS", *IEEE ISSCC*, pp. 84-85, 1994.