Minimizing Power Consumption in Digital CMOS Circuits

ANANTHA P. CHANDRAKASAN AND ROBERT W. BRODERSEN, FELLOW, IEEE

An approach is presented for minimizing power consumption for digital systems implemented in CMOS which involves optimization at all levels of the design. This optimization includes the technology used to implement the digital circuits, the circuit style and topology, the architecture for implementing the circuits and at the highest level the algorithms that are being implemented. The most important technology consideration is the threshold voltage and its control which allows the reduction of supply voltage without significant impact on logic speed. Even further supply reductions can be made by the use of an architecture-based voltage scaling strategy, which uses parallelism and pipelining, to tradeoff silicon area and power reduction. Since energy is only consumed when capacitance is being switched, power can be reduced by minimizing this capacitance through operation reduction, choice of number representation, exploitation of signal correlations, resynchronization to minimize glitching, logic design, circuit design, and physical design. The low-power techniques that are presented have been applied to the design of a chipset for a portable multimedia terminal that supports pen input, speech I/O and full-motion video. The entire chipset that performs protocol conversion, synchronization, error correction, packetization, buffering, video decompression and D/A conversion operates from a 1.1 V supply and consumes less than 5 mW.

I. INTRODUCTION

In recent years, the desirability of portable operation of all types of electronic systems has become clear and a major factor in the weight and size of portable devices is the amount of batteries which is directly impacted by the power dissipated by the electronic circuits. In addition, the cost of providing power (and associated cooling) has resulted in significant interest in power reduction even in nonportable applications which have access to a power source. In spite of these concerns, 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 approach which will be presented here, takes another viewpoint, in which all possible aspects of a system design are investigated with the goal of reducing the power consumption. These considerations range from the technology being used for the implementation, the circuit and logic topologies, the digital architectures and even the algorithms being implemented. What is assumed is that the application, which is desired to be implemented with low power is known, and tradeoffs can be made as long as the functionality required of this application is met within a given time constraint.

Maintaining a given level of computation or throughput is a common concept in signal processing and other dedicated applications, in which there is no advantage in performing the computation faster than some given rate, since the processor will simply have to wait until further processing is required. This is in contrast to general purpose computing, where the goal is often to provide the fastest possible computation without bound. One of the most important ramifications of only maintaining throughput is that it enables an architecture driven voltage scaling strategy, in which aggressive voltage reduction is used to reduce power, and the resulting reduction in logic speed is compensated through parallel architectures to maintain throughput. However, the techniques presented are also applicable to the general purpose environment, if the figure of merit is the amount of processing per unit of power dissipation (e.g., MIPS/W). Since in this case the efficiency in implementing the computation is considered and voltage scaling decreases the energy expended per evaluation.

The optimization to minimize area at all costs, has only been secondary to the fixation on increasing circuit speed and again our position is that this should be examined with respect to its effect on power consumption. Some of the techniques that will be presented will come at the expense of increased silicon area and thus the cost of the implementation will be increased. The desirability of this tradeoff can only be determined with respect to a given market situation, but in many cases a moderate increase in area can have substantial impact on the power requirements. It is clear that if power reduction is more important than increasing circuit clock rate, then the area consumed by large clock buffers, power distribution busses and predictive circuit architectures would be better spent to reduce the power dissipation.

Manuscript received May 1, 1994; revised July 24, 1994. This work was supported by ARPA. A. Chandrakasan's work was supported by an IBM fellowship.

A. Chandrakasan is with the Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

R. W. Brodersen is with the Department of EECS, University of California, Berkeley, CA 94720 USA.

IEEE Log Number 9408188.

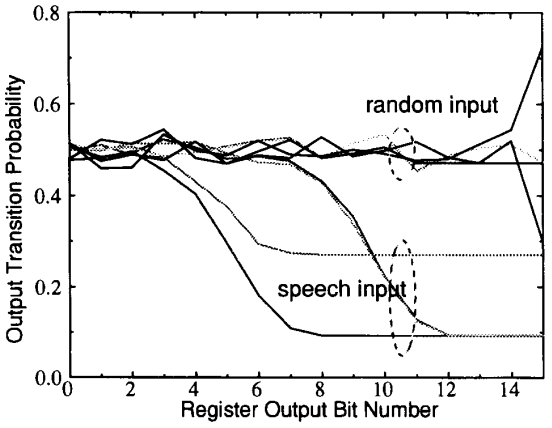


Fig. 1. Dependence of activity on statistics: Correlated versus random input.

Since CMOS circuits do not dissipate power if they are not switching, a major focus of low power design is to reduce the switching activity to the minimal level required to perform the computation. This can range from simply powering down the complete circuit or portions of it, to more sophisticated schemes in which the clocks are gated or optimized circuit architectures are used which minimize the number of transitions. An important attribute which can be used in circuit and architectural optimization is the correlation which can exist between values of a temporal sequence of data, since switching should decrease if the data is slowly changing (highly correlated). An example of the difference in the number of transitions which can be obtained for a highly correlated data stream (human speech) versus random data is shown in Fig. 1—the transition activity for a few registers in an FIR filter design. For an architecture which does not destroy the data correlation, the speech data switches 80% less capacitance than the random input. In addition, the sequencing of operations can result in large variations of the switching activity due to these temporal correlations.

After a brief summary of the sources of power dissipation in CMOS circuits, techniques will be presented to reduce power dissipation at the four levels of CMOS system design: technology, circuits, architectures, and algorithms. This discussion will be followed by an application of these techniques to a demanding multimedia application.

II. Factors Influencing CMOS Power Consumption

There are three major sources of power dissipation in digital CMOS circuits which are summarized in the following equation [1]:

$$P_{\text{avg}} = P_{\text{switching}} + P_{\text{short-circuit}} + P_{\text{leakage}} = \alpha_{0\to1} C_L \cdot V_{dd}^2 \cdot f_{clk} + I_{sc} \cdot V_{dd} + I_{\text{leakage}} \cdot V_{dd}. \quad (1)$$

The first term represents the switching component of power, where C_L is the load capacitance, f_{clk} is the clock frequency and $\alpha_{0\to1}$ is the node transition activity factor (the average number of times the node makes a power conusming transition in one clock period). The second term is due to the direct-path short circuit current, I_{sc}, which arises when both the NMOS and PMOS transistors are simultaneously active, conducting current directly from supply to ground. Finally, leakage current, I_{leakage}, which can arise from substrate injection and subthreshold effects, is primarily determined by fabrication technology considerations.

A. Capacitive Voltage Transitions

The switching or dynamic component of power consumption arises when the capacitive load, C_L, of a CMOS circuit is charged through PMOS transistors to make a voltage transition from 0 to the high voltage level, which is usually the supply, V_{dd}. For an inverter circuit, the energy drawn from the power supply for this positive going transition is $C_L V_{dd}^2$, half of which is stored in the output capacitor and half is dissipated in the PMOS device. On the V_{dd} to 0 transition at the output, no charge is drawn from the supply, however the energy stored in the capacitor ($\frac{1}{2} C_L V_{dd}^2$) is dissipated in the pull-down NMOS device. If these transitions occur at a clock rate, f_{clk}, the power drawn from the supply is $C_L V_{dd}^2 f_{clk}$. However, in general, the switching will not occur at the clock rate (except for clock buffers), but rather at some reduced rate which is best described probabilistically. $\alpha_{0\to1}$ is defined as the average number of times in each clock cycle that a node with capacitance, C_L, will make a power consuming transition (0 to 1), resulting in an average switching component of power for a CMOS gate to be,

$$P_{\text{switching}} = \alpha_{0\to1} C_L V_{dd}^2 f_{clk}. \quad (2)$$

Since the energy expended for each switching event in CMOS circuits is $C_L V_{dd}^2$, it has the extremely important characteristic that it becomes quadratically more efficient as the high transition voltage level is reduced. An effective capacitance, $C_{\text{effective}}$, can be defined which includes the transition activity occurring and is equal to $\alpha_{0\to1} C_L$, yielding an average transition energy of $C_{\text{effective}} V_{dd}^2$. Clearly, after the high transition level is reduced by the maximum extent possible, the next step is to minimize $C_{\text{effective}}$, through the choice of logic function, logic style, circuit topology, data statistics, and the sequencing of operations. In addition, an optimization can be made both statically, in which the probabilities of transition are made with respect to the Boolean operation and dynamically which includes the effect of spurious "glitching" transitions which can occur before the logic settles out to its static value.

1) Logic Function: The type of logic function, NAND, NOR or a more complex gate, will all yield different static transition probabilities. Let us first consider a static 2-input NOR gate and assume that only one input transition is possible during a clock cycle and also assume that the inputs to the NOR gate have a uniform input distribution of high and low levels. This means that the four possible states for inputs A and B (00, 01, 10, 11) are equally likely. For a

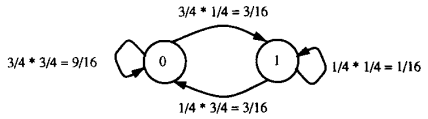


Fig. 2. State transition diagram for a 2 input NOR gate assuming random inputs.

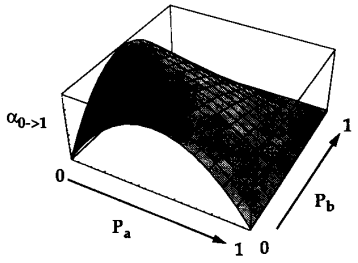


Fig. 3. Transition probability for a two input NOR gate as a function of input statistics.

NOR gate, the probability that the output is **0** is $\frac{3}{4}$ and that it will be **1** is $\frac{1}{4}$. The 0 to 1 transition probability of a CMOS gate is given by the probability that the output is in the **0** ($=\frac{3}{4}$) state multiplied by the probability the next state is **1** ($=\frac{1}{4}$). For a NOR gate this translates to

$$\alpha_{0\to1} = p(0)p(1) = p(0)(1-p(0)) = \tfrac{3}{4}(1\tfrac{3}{4}) = \tfrac{3}{16}. \quad (3)$$

The state transition diagram annotated with transition probabilities is shown in Fig. 2. Note that the output probabilities are no longer uniform.

For a 2-input static XOR gate on the other hand, the $0 \to 1$ transition probability once again assuming uniformly distributed inputs is given by $p(0)(1-p(0)) = \frac{1}{2}(1-\frac{1}{2}) = \frac{1}{4}$.

2) Logic Style: One basic choice of logic style is between static and dynamic CMOS. For a dynamic implementation of the 2 input NOR gate which is implemented as an NMOS-tree with PMOS precharge, power is consumed during the precharge operation for the times when the output capacitor was discharged the previous cycle. Since all inputs are equi-probable, there is then a 75% probability that the output node will discharge immediately after the precharge phase, yielding a transition probability of 0.75. The corresponding probability is smaller for a static implementation (as computed in the previous subsection). Note that for the dynamic case, the activity depends only on the signal probability, while for the static case the transition probability depends on previous state. If the inputs to a static CMOS gate do not change from the previous sample period, then the gate does not switch. This is not true in the case of dynamic logic in which gates can switch. For a dynamic NAND gate, the activity is $\frac{1}{4}$ (since there is a 25% chance that the output will be discharged) while it is $\frac{3}{16}$ for a static implementation. Other considerations with respect to logic style is the amount of capacitance on the switching node and the sensitivity to supply voltage reduction.

3) Signal Statistics: In the previous sections, the NOR gate was analyzed assuming random inputs. However, signals in a circuit are typically not always random and

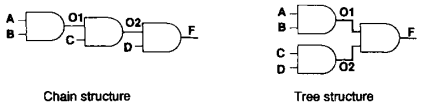


Fig. 4. Simple example to demonstrate the influence of circuit topology on activity.

Table 1 Probabilities for Tree and Chain Toplologies

	O1	O2	F
P_1 (chain)	1/4	1/8	1/16
$P_0 = 1 - P_1$ (chain)	3/4	7/8	16/16
$P_{0\to1}$ (chain)	3/16	**7/64**	15/256
P_1 (tree)	1/4	1/4	1/16
$P_0 = 1 - P_1$ (tree)	3/4	3/4	15/16
$P_{0\to1}$ (tree)	3/16	**3/16**	15/256

it is necessary to consider the effect of signal statistics on power. To illustrate the influence of signal statistics on power consumption, consider once again a 2 input NOR gate, and let P_a and P_b be the probabilities that the inputs A and B are **1**. In this case the probability that the output node is a **1** is given by:

$$P_1 = (1-P_a)(1-P_b). \quad (4)$$

Therefore, the probability of transitioning from 0 to 1 is

$$\alpha_{0\to1} = P_0P_1 = (1-(1-P_a)(1-P_b))(1-P_a)(1-P_b). \quad (5)$$

Fig. 3 shows the transition probability as a function of P_a and P_b. From this plot, it is clear that understanding the signal statistics and their impact on switching events can be used to significantly impact the power dissipation.

4) Circuit Topology: The manner in which logic gates are interconnected can have a strong influence on the overall switching activity. There are two components to switching activity: a static component (which does not take into account the timing behavior and is strictly a function of the topology and the signal statistics) and a dynamic component (which takes into account the timing behavior of the circuit). To illustrate this point consider two alternate implementations of $F = A \cdot B \cdot C \cdot D$ as shown in Fig. 4. First consider the static behavior assuming that all primary inputs (A, B, C, D) are uncorrelated and random (i.e., $P_{1(a,b,c,d)} = 0.5$). For an AND gate, the probability that the output is in the **1** state is given by

$$P_1 = P_aP_b. \quad (6)$$

Therefore the probability that the output will make a 0 to 1 transition is given by

$$\alpha_{0\to1} = P_0P_1 = (1-P_1)P_1 = (1-P_aP_b)P_aP_b. \quad (7)$$

Given this, the signal and transition probabilities can be computed for the two topologies shown in Fig. 4 which is summarized in Table 1.

The results indicate that the chain implementation will have an overall lower switching activity than the tree implementation for random inputs. However, looking strictly at the static behavior of the circuit is not adequate,

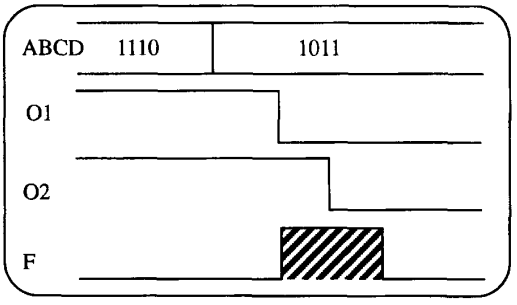


Fig. 5. Circuit imbalances result in spurious transitions.

and it is also important to consider the timing behavior to accurately make power tradeoffs of various topologies. Timing skew between signals can cause spurious transitions (also called glitches) resulting in extra power. To illustrate this tradeoff, consider once again the two topologies in Fig. 4. For the chain implementation with an input transition of 1110 → 1011 for $ABCD$, assuming a unit delay for each gate, ignoring dynamic switching effects, the output node will not make a transition. However, due to timing skew through the logic, the output will make an "extra" transition; i.e., O2 will only be valid 2 units after the inputs arrive, causing the output AND gate to evaluate with the new input D and the previous value of O2 (Fig. 5). The tree implementation, on the other hand, is balanced and is glitch free. This example demonstrates that a circuit topology can have a smaller static component of activity while having a higher dynamic component.

The actual waveforms illustrating the glitching behavior in static CMOS circuits are shown in Fig. 6, which is the SPICE simulation of a static 16-b adder, with all bits of IN0 and the CIN of the LSB going from "zero" to "one," and with all the bits of IN1 set to "zero." For all bits, the resultant sum should be zero; however, the propagation of the carry signal causes a "one" to appear briefly at most of the outputs. These spurious transitions dissipate extra power over that strictly required to perform the computation. Note that some of the bits only have partial glitching. The number of these extra transitions is a function of input patterns, internal state assignment in the logic design, delay skew, and logic depth. Though it is possible with careful logic design to eliminate these transitions (for example using balanced paths as described in Section V-D), dynamic logic intrinsically does not have this problem, since any node can undergo at most one power-consuming transition per clock cycle.

B. Short-Circuit Component of Power

The previous section analyzed the switching component of power consumption which corresponds to the amount of energy required to charge parasitic capacitors. The switching component of power is independent of the rise and fall times at the input of logic gates. Finite rise and fall times of the input waveforms however result in a direct current path between V_{dd} and GND which exist for a short period of time during switching. Specifically, when

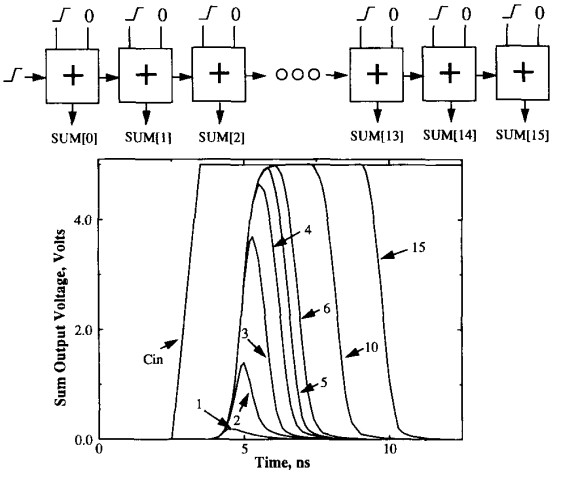


Fig. 6. Waveforms for a 16-b adder demonstrating glitching behavior.

the condition $V_{tn} < V_{\text{in}} < V_{dd} - |V_{tp}|$ holds for the input voltage, where V_{tn} and V_{tp} are NMOS and PMOS threshold voltages, there will be a conductive path open between V_{dd} and GND because both the NMOS and PMOS devices will be simultaneously on. Short circuit currents are significant when the rise/fall time at the input of a gate is much larger than the output rise/fall time. This is because the short-circuit path will be active for a longer period of time. To minimize the total average short-circuit current, it is desirable to have equal input and output edge times [2]. In this case, the power consumed by the short-circuit currents is typically less than 10% of the total dynamic power. An important point to note is that if the supply is lowered to be *below* the sum of the thresholds of the transistors, $V_{dd} < V_{Tn} + |V_{Tp}|$, the short-circuit currents can be eliminated because both devices will not be on at the same time for any value of input voltage. As will be shown later, the architecture driven voltage scaling that is proposed allows supply voltages which satisfy this criteria.

C. Leakage Component of Power

There are two types of leakage currents: reverse-bias diode leakage on the transistor drains and subthreshold leakage through the channel of an off device.

The diode leakage occurs when a transistor is turned off and another active transistor charges up/down the drain with respect to the former's bulk potential. For example, consider an inverter with a high input voltage, in which the NMOS transistor is turned on and the output voltage is driven low. The PMOS transistor will be turned off, but its drain-to-bulk voltage will be equal to $-V_{dd}$ since the output voltage is at 0 V and the bulk for the PMOS is at V_{dd}. The resulting current will be approximately $I_L = A_D J_S$, where A_D is the area of the drain diffusion, and J_S is the leakage current density, set by the technology and weakly dependent on the supply voltage. For a typical CMOS process, J_S is approximately 1–5 pA/μm^2 (25° C), and the minimum A_D is 7.2 μm^2

for a 1.2 μm minimum feature size. J_S doubles with every 9° increase in temperature. For a 1 million transistor chip, assuming an average drain area of 10 μm^2, the total leakage current is on the order of 25 μA. While this is typically a small fraction of the total power consumption in most chips, it could be significant for a system application which spends much of its time in standby operation, since this power always being dissipated even when no switching is occurring.

The second component of the leakage power is the subthreshold leakage which occurs due to carrier diffusion between the source and the drain when the gate-source voltage, V_{gs}, has exceeded the weak inversion point, but is still below the threshold voltage V_t, where carrier drift is dominant [3]. In this regime, the MOSFET behaves similarly as a bipolar transistor, and the subthreshold current is exponentially dependent on the gate-source voltage V_{gs}. The current in the subthreshold region is given by

$$I_{ds} = \kappa e^{(V_{gs}-V_t)/(nV_T)}(1 - e^{V_{ds}/V_T}) \tag{8}$$

where κ is a function of the technology, V_T is the thermal voltage (KT/q) and V_t is the threshold voltage. For $V_{ds} \gg V_T$, $(1-e^{-Vds/VT}) \approx 1$; that is, the drain to source leakage current is independent of the drain-source voltage V_{ds}, for V_{ds} approximately larger than 0.1 V.

Associated with this is the subthreshold slope S_{th}, which is the amount of voltage required to drop the subthreshold current by one decade. The subthreshold slope can be determined by taking the ratio of two points in this region (from (8)):

$$\frac{I_1}{I_2} = e^{(V_1-V_2)/(nV_T)} \tag{9}$$

which results in $S_{th} = nV_T \ln$ (10). At room temperature, typical values for S_{th} lie between 60–90 mV/(decade current), with 60 mV/dec being the lower limit. Clearly, the lower S_{th} is, the better, since it is desirable to have the device "turnoff" as close to V_t as possible and this is one important advantage of silicon-on-insulator technologies which have an S_{th} near 60 mV [4]. The choice of optimal V_t for low voltage applications that trades-off subthreshold leakage and switching currents is presented in Section III.

III. Technology Optimization

As noted in (1), the energy per transition is proportional to V_{dd}^2. Therefore, it is only necessary to reduce the supply voltage for a quadratic improvement in the power-delay product of CMOS logic. Unfortunately, we pay a speed penalty for a V_{dd} reduction (as seen from Fig. 7), with the delays drastically increasing as V_{dd} approaches the threshold voltages of the devices. Even though the exact analysis of the delay is quite complex if the nonlinear characteristic of a CMOS gate are taken into account, it is found that a simple first-order derivation adequately predicts the experimentally determined dependence and is given by

$$T_d = \frac{C_L \times V_{dd}}{I} = \frac{C_L \times V_{dd}}{\frac{\mu C_{ox}}{2}(W/L)(V_{dd} - V_t)^2}. \tag{10}$$

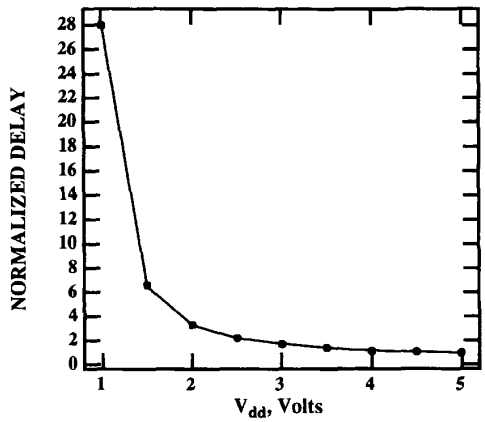


Fig. 7. Normalized delay versus V_{dd} for a typical gate in a standard CMOS process.

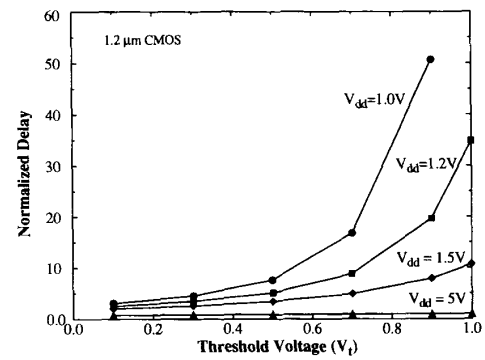


Fig. 8. Effect of threshold reduction on the delay for various supply voltages.

Since the objective is to reduce power consumption while keeping the throughput of overall system fixed, compensation for these increased delays at low voltages is required. In Section V-A, an architecture driven voltage scaling strategy is presented in which parallel and pipelined architectures are used to compensate for the increased gate delays at reduced supply voltages and meet throughput constraints. Another approach to reduce the supply voltage without loss in throughput is to modify the threshold voltage of the devices. Low threshold voltage devices have been used to implement inverter circuits at supply voltages as low as 200 mV [5]. Reducing the threshold voltage allows the supply voltage to be scaled down (and therefore lower switching power) without loss in speed. For example, a circuit running at a supply voltage of 1.5 V with $V_t = 1$ V will have approximately the same performance as the circuit running at a supply voltage of 0.9 V and a $V_t = 0.5$ V using the simple first order theory of (10). Fig. 8 shows a plot of normalized delay versus threshold voltage for various supply voltages.

Since a significant power improvement can be gained through the use of low-threshold MOS devices, the question of how low the thresholds can be reduced must be addressed. The limit is set by the requirement to retain

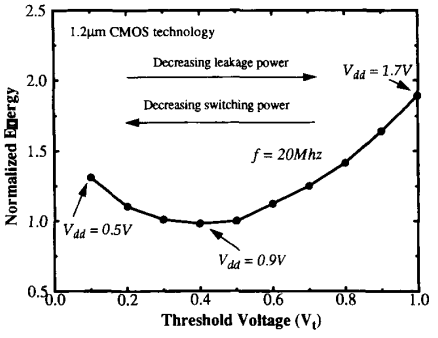


Fig. 9. Compromise between dynamic and leakage power dissipation through V_t variation.

adequate noise margins and the increase in subthreshold currents. Noise margins will be relaxed in low power designs because of the reduced currents being switched, however, the subthreshold currents can result in significant static power dissipation.

Fig. 9 shows a plot of energy versus threshold voltages for a fixed throughput for a 16-b 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 obtained at a 20 Mhz 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 subthreshold leakage.

IV. Physical, Circuit, and Logic Level Optimizations

The next level of system design involves optimizing the physical, circuit and logic levels. A few examples which illustrate the tradeoffs that can be made at this level of system design are presented, which include optimizing place and route, transistor sizing, reduced swing logic, logic minimization, and logic level power down. Support circuitry for low voltage operation, which includes voltage level converters and low-voltage dc/dc converters, will be presented.

A. Place and Route Optimization

At the layout level, the place and route should be optimized such that signals that have high switching activity (such as clocks) should be assigned short wires and signals with lower switching activities can be allowed progressively longer wires. Current design tools typically minimize the overall area or wire lengths given a timing constraint, which does not necessarily reduce the overall capacitance switched. An example of the benefits of this optimization is presented in Section VII-A.

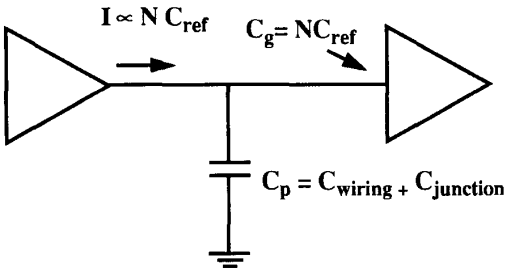


Fig. 10. Circuit model for analyzing the effect of transistor sizing.

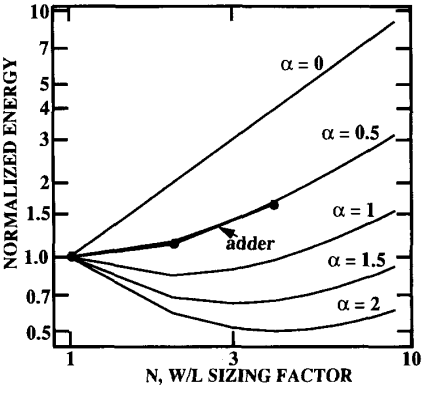


Fig. 11. Plot of energy versus transistor sizing factor for various parasitic contributions.

B. Transistor Sizing

Independent of the choice of logic family or topology, optimized transistor sizing will play an important role in reducing power consumption. For low power, as is true for high speed design, it is important to equalize all delay paths so that a single critical path does not unnecessarily limit the performance of the entire circuit. However, beyond this constraint, there is the issue of what extent the (W/L) ratios should be uniformly raised for all the devices, yielding a uniform decrease in the gate delay and hence allowing for a corresponding reduction in voltage and power. It is shown in this section, that if voltage is allowed to vary, that the optimal sizing for low power operation is quite different from that required for high speed.

In Fig. 10, a simple two-gate circuit is shown, with the first stage driving the gate capacitance of the second, in addition to the parasitic capacitance C_p due to substrate coupling and interconnect. Assuming that the input gate capacitance of both stages is given by NC_{ref}, where C_{ref} represents the gate capacitance of a MOS device with the smallest allowable (W/L), then the delay through the first gate at a supply voltage V_{ref} is given by

$$T_N = K\frac{(C_p + NC_{\text{ref}})}{(NC_{\text{ref}})}\frac{V_{\text{ref}}}{(V_{\text{ref}} - V_t)^2} = K(1 + \alpha/N)\frac{V_{\text{ref}}}{(V_{\text{ref}} - V_t)^2} \quad (11)$$

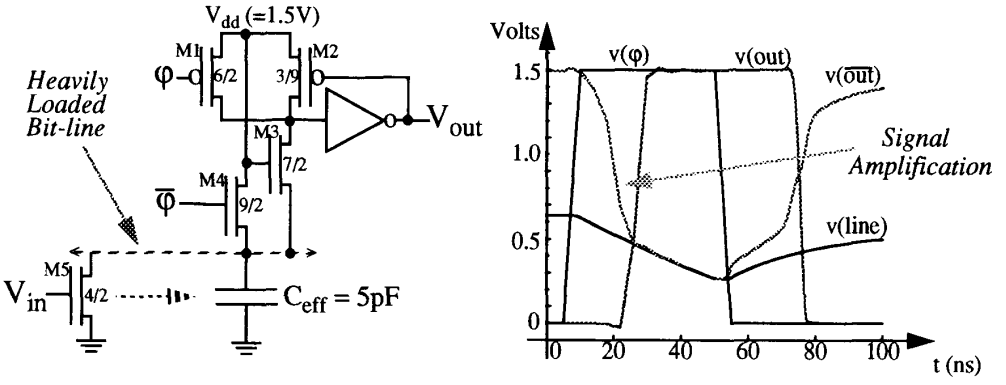


Fig. 12. Signal amplification/swing reduction in memory circuits.

where α is defined as the ratio of C_p to C_{ref}, and K represents terms independent of device width and voltage. For a given supply voltage V_{ref}, the speed up of a circuit whose W/L ratios are sized up by a factor of N over a reference circuit using minimum size transistors ($N = 1$) is given by $(1 + \alpha/N)/(1 + \alpha)$. In order to evaluate the energy performance of the two designs at the same speed, the voltage of the scaled solution is allowed to vary as to keep delay constant. Assuming that the delay scales as $1/V_{dd}$ (ignoring threshold voltage reductions in signal swings) the supply voltage, V_N, where the delay of the scaled design and the reference design are equal is given by

$$V_N = \frac{(1 + \alpha/N)}{(1 + \alpha)} V_{\text{ref}}. \tag{12}$$

Under these conditions, the energy consumed by the first stage as a function of N is given by

$$\begin{aligned} \text{Energy}(N) &= (C_p + NC_{\text{ref}})V_N^2 \\ &= \frac{NC_{\text{ref}}(1 + \alpha/N)^3 V_{\text{ref}}^2}{(1 + \alpha)^2}. \end{aligned} \tag{13}$$

After normalizing against E_{ref} (the energy for the minimum size case), Fig. 11 shows a plot of Energy(N)/Energy(1) versus N for various values of α. When there is no parasitic capacitance contribution (i.e., $\alpha = 0$), the energy increases linearly with respect to N, and the solution utilizing devices with the smallest (W/L) ratios results in the lowest power. At high values of α, when parasitic capacitances begin to dominate over the gate capacitances, the power decreases temporarily with increasing device sizes and then starts to increase, resulting in a optimal value for N. The initial decrease in supply voltage achieved from the reduction in delays more than compensates the increase in capacitance due to increasing N. However, after some point the increase in capacitance dominates the achievable reduction in voltage, since the incremental speed increase with transistor sizing is very small (this can be seen in (11), with the delay becoming independent of α as N goes to infinity). Throughout the analysis we have assumed that the parasitic capacitance is independent of device sizing, which is the common case when interconnect dominates. However, the drain and source diffusion and perimeter capacitances actually increase with increasing area, favoring smaller size devices and making the above a worst-case analysis. Also plotted in Fig. 11 are simulation results from extracted layouts of an 8-b adder carry chain for three different device (W/L) ratios (N =1, $N = 2$, and $N = 4$). The curve follows the simple first-order model derived very well, and suggests that this example is dominated more by the effect of gate capacitance rather than parasitics. In this case, increasing devices (W/L)'s does not help, and the solution using the smallest possible (W/L) ratios results in the best sizing.

These results indicate that minimum sized devices should be used when the total load capacitance is not dominated by the interconnect. This is indeed the case for circuitry that drive local busses inside datapaths. To drive large capacitances, such as between datapath modules or off-chip, buffers with appropriate strength can be used.

A low-power cell library has been designed which mostly uses minimum sized devices to implement most logic functions and has been used to design the circuits described in Section VII. This cell library typically switches a factor of 2–3 lower capacitance compared to cell libraries optimized for speed.

C. Reduced Swing Logic

Reducing the power supply voltage is clearly a very effective way to reduce the energy per operation since it has a quadratic impact on the power consumption. At a given reduced supply voltage, the output of CMOS logic gates will make rail to rail transitions; an approach to reducing the power consumption further at a fixed reduced supply voltage is to reduce the swing on the output node. For example, using an NMOS device to pull up the output will limit the swing to $V_{dd} - V_t$, rather than rising all the way to the supply voltage. The power consumed for a $0 \rightarrow V_{dd} - V_t$ transition will be $C_L V_{dd}(V_{dd} - V_t)$, and therefore the power consumption reduction over a rail to rail scheme will be $\propto V_{dd}/(V_{dd} - V_t)$. This scheme of using an NMOS device to reduce the swing has two important negative consequences: First, the noise margin for output high (NM_H) is reduced by the amount V_T, which can reduce the margin to 0 V, if the supply voltage is set near the sum of the thresholds.

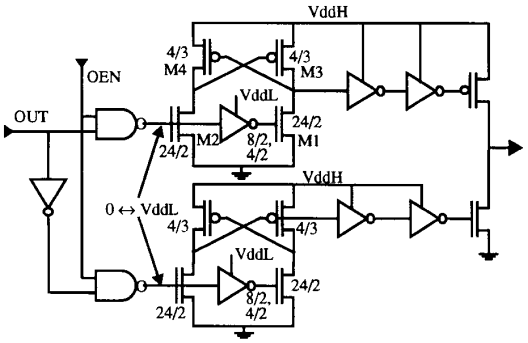


Fig. 13. Level conversion output pad driver.

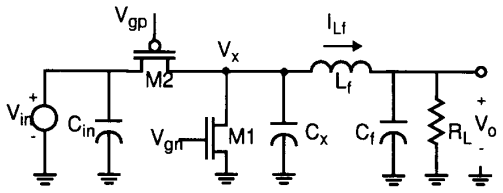


Fig. 14. Low-voltage dc/dc buck converter.

Second, since the output does not rise to the upper rail, a static gate connected to the output can consume static power for a high output voltage (since the PMOS of the next stage will be "on"), increasing the effective energy per transition.

Therefore, to utilize the voltage swing reduction, special gates are needed to restore the noise margin to the signal, and eliminate short-circuit currents. These gates require additional devices that will contribute extra parasitic capacitances. Fig. 12 shows a simplified schematic of such a gate, used in the FIFO memory cells of a low-power cell library [6]. This circuit uses a precharged scheme and the device M3 is used to clip the voltage of the bit-line (which has several transistors similar to M5 connected to it) to $V_{dd}-V_t$, where $V_t > V_{t0}$ due to the body effect. The devices M1 and M4 are used to precharge the internal node (the input of the inverter) to V_{dd}, and the bit-line to $V_{dd}-V_T$. During evaluation (φ = "1"), if V_{in} is high, the bit-line will begin to drop, as shown in the SPICE output next to the schematic. Because the capacitance ratio of the bit-line to the internal node is very large, once the bit-line has dropped roughly 200 mV to sufficiently turn on M3, the internal node quickly drops to the potential of the bit-line, providing signal amplification. Thus this circuit greatly reduces the voltage swing on the high-capacitance bit-line, which reduces the energy, and provides signal amplification, which reduces the delay, as well. If the load on the bit-line was on the order of just a few gates, the energy savings would be marginal, due to the extra parasitic capacitances and therefore the signal swing reduction technique presented above is most useful for high-capacitance nodes.

D. Low-Voltage Support Circuitry: Level Converting Pad

One important type of circuit required for low-voltage operation is a level-shifter that can convert low-voltage signal swings from the core of the low-power chips (e.g., 1.5 V) to the high-voltage signal swings required by I/O devices (e.g., 5 V required by displays) or vice-versa. Fig. 13 shows the schematic of a level converting output pad driver.

This tri-stateable output buffer uses two supply voltages, the low-voltage supply, $V_{dd}L$, that is tied to the pad-ring and the high-voltage supply, $V_{dd}H$, coming in through another unbuffered pad. The low-to-high conversion circuit is a PMOS cross coupled pair (M3, M4) connected to the high supply voltage, driven differentially (via M1, M2) by the low-voltage signal from the core. The N-device pulldowns, M1 and M2, are dc ratioed against the cross coupled P-device pullups, M3 and M4, so that a low-swing input ($V_{dd}L = 1$ V) guarantees a correct output transition ($V_{dd}H = 5$ V). That is, the PMOS widths are sized so that the drive capability of the NMOS can overpower the drive of the PMOS, and reverse the state of the latch. This level-converting pad consumes power only during transitions and it consumes no dc power. The remaining buffer stages and output driver are supplied by $V_{dd}H$. This level-conversion pad will work only with an NWELL process since it requires isolated wells for the PMOS devices that are connected to the high voltage and the ones that are connected to the low-voltage devices.

E. Low-Voltage Support Circuitry: High Efficiency Low-Voltage dc/dc Conversion

The architecture driven voltage scaling strategy which will be presented later assumes that the supply voltage is a free variable. In order to realize such portable systems in which different parts of the system could operate at their own "optimum" supply voltage and communicate with each other using the level-conversion circuitry described in Section IV-D, the design of high efficiency low-voltage (in which the voltage can be made programmable) switching regulators must be considered. Fig. 14 shows the schematic of a Buck regulator that can be used to generate low-supply voltages with high efficiency. The converter works by chopping the input voltage to reduce the average voltage. This produces a square-wave of duty cycle D with frequency f_s at the inverter output node V_x. The chopped signal is filtered by the second-order low-pass filter (L_f and C_f) to reduce the ac component to an acceptable ripple value. Buck converters are capable of a 0% to 100% duty factor and the output voltage is approximately equal to the input voltage multiplied by duty factor. For example, with $V_{\text{in}} = 6$ V and $D = 25\%$, the output voltage is 1.5 V. The output can be set to an arbitrary value by controlling the duty cycle on the inverter output node V_x. There are three main sources of dissipation that cause the conversion efficiency of this circuit to be less than unity: conduction loss (I^2R loss in the power transistors and filter components), switching loss ($C_xV_{\text{in}}^2f_s$ loss due to parasitics), and $C_gV_{\text{in}}^2f_s$ gate drive loss for each FET. Four techniques are presented in [7] to improve the efficiency of the converter at reduced voltages and current levels. Synchronous Rectification: The free-wheeling diode used

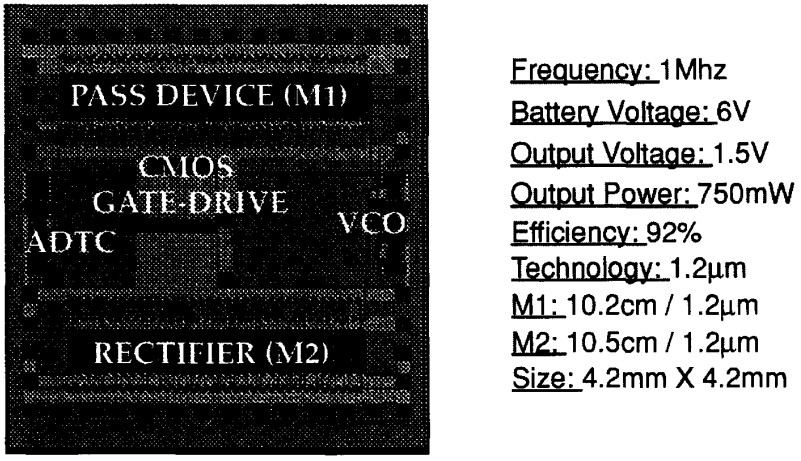


Fig. 15. Die photo of a 1.5 V buck converter [7].

in conventional high voltage converters is replaced with a gated NMOS device (M1). For low output voltage levels, the diode's voltage drop becomes a significant fraction of the output voltage, leading to large conduction loss. A gated NMOS device can achieve the same function more efficiently.

1) Soft Switching: Capacitive switching loss is nearly eliminated by using the filter inductor as a current source to charge and discharge the inverter node; therefore, the power transistors are turned on and off at $V_{DS} = 0$ (referred to as zero voltage switching (ZVS)). This factor is especially important for high-frequency converters.

2) Adaptive Dead Time Control: ZVS depends critically on the dead-time, during which neither M1 nor M2 conducts. A fixed dead-time designed assuming an average current load can result in significant losses if operating conditions change. An adaptive dead-time control through a negative feedback loop is used to accommodate varying operating conditions and process variations.

3) Resonant-Transition Gate-drive: A resonant inductor is used to charge and discharge the gate capacitances of the power devices, recycling a significant fraction of the gate charge. Fig. 15 shows the die photo of the low-voltage converter.

F. Logic Minimization and Technology Mapping

Recently some effort has gone into changing the optimization criteria for synthesizing logic structures (both combinational and sequential) to one that addresses low power. Typical reduction in power consumption that can be achieved by optimizing at this level for random logic is on the order of 25%. Techniques have been proposed which choose logic to minimize switching [8], position registers through retiming to reduce glitching activity [9] and to decrease power during the technology mapping phase [10], [11] choosing gates from a library which reduces switching; for example a three input AND gate can be implemented as a single 3-input gate (i.e., NAND followed by an inverter) or two 2-input AND gates with different power results.

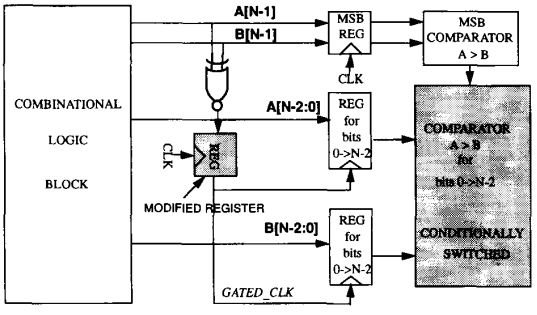


Fig. 16. Using gated clocks to reduce power.

G. Logic Level Power-Down and Gated Clocks

Powering down has traditionally been applied only at the chip and module levels, however, the application to the logic level can also be very beneficial to reduce the switching activity at the expense of some additional control circuitry [12]. Assume a pipelined system for comparing the output of two numbers from a block of combinational logic as shown in Fig. 16; the first pipeline stage is a combinational block and the next pipeline stage is a comparator which performs the function $A > B$, where A and B are generated in the first stage (i.e., from the combinational block). If the most significant bits, $A[N-1]$ and $B[N-1]$, are different then the computation of $A > B$ can be performed strictly from the MSB's and therefore the comparator logic for bits $A[N-1:0]$ and $B[N-2:0]$ is not required (and hence the logic can be powered down). If the data is assumed to be random (i.e., there is a 50% chance that $A[N-1]$ and $B[N-1]$ are different), the power savings can be quite significant. One approach to accomplish this is to gate the clocks as shown in Fig. 16. The XNOR output of the $A[N-1]$ and $B[N-1]$ is latched by a special register to generate a gated clock. This gated clock is then used to clock the lower order registers. Since the latch to the lower bits is conditional, they must be made static. The standard TSPC register [13] is shown in Fig. 17 as modified

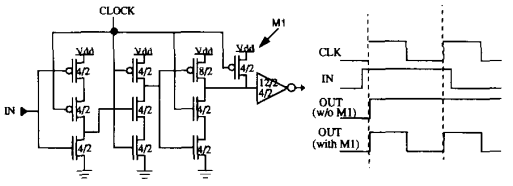


Fig. 17. Schematic of a modified TSPC latch that is used to generate gated clocks.

to support clock gating. As shown in the timing on the right side of Fig. 17, with the extra PMOS device M1, the output can be forced to a ZERO during the low phase of the clock. Without this device, it will not be possible to generate rising edges for the gated clock on two consecutive rising edges of the system clock. Gated clocks have been used extensively in the system presented in Section VII.

V. Architecture Level Optimization

There is a great deal of freedom in optimizing architectures for low power. An architecture driven voltage scaling strategy is presented in this section in which concurrent architectures are used to retain throughput at reduced supply voltages. This is an extremely effective approach for power consumption reduction and can in many cases, yield more than an order of magnitude savings. Various architectural techniques are also presented to reduce the effective capacitance switched which involves the choice of number representation, exploitation of signal correlations, and resynchronization to minimize glitching.

A. Architecture Driven Voltage Scaling

The simple first order delay analysis presented in (10) is reasonably accurate for long channel devices. However, as feature sizes shrink below 1.0 μm, the delay characteristics as a function of lowering the supply voltage deviate from the first order theory presented since it does not consider carrier velocity saturation under high electric fields. As a result of velocity saturation, the current is no longer a quadratic function of the voltage but linear; hence, the current drive is significantly reduced and is approximately given by $I = WC_{ox}(V_{dd} - V_t)\nu_{\max}$ [14]. Given this and assuming the delay of a circuit is given by CV_{dd}/I, we see that the delay for submicron circuits is relatively independent of supply voltages at high electric fields. A "technology" based approach proposes choosing the power supply voltage based on maintaining the speed performance for a given submicron technology [15]. By exploiting the relative independence of delay on supply voltage at high electric fields, the voltage can be dropped to some extent for a velocity-saturated device with very little penalty in speed performance. Because of this effect, there is some movement to a 3.3 V industrial voltage standard since at this level of voltage reduction, to the "critical voltage," there is not a significant loss in circuit speed [15]. This approach can achieve a 60% reduction in power when compared to a 5 V operation [16].

The above mentioned "technology" based approach is focusing on reducing the voltage while maintaining device speed, instead of attempting to maintain computational throughput at a minimum power level. From (1) it is clear that CMOS logic gates achieve lower power-delay products as the supply voltages are reduced. In fact, once a device is in velocity saturation there is a further degradation in the energy per computation, so in minimizing the energy required for computation, the above critical voltage provides an *upper* bound on the supply voltage (whereas in the technology scaling theory it provided a *lower* bound!). It now will be the task of the architecture to compensate for the reduced circuit speed, that comes with operating below the critical voltage. Examples of optimizing architectures for both arithmetic and memory computation are presented below.

1) Arithmetic Computation: A Simple Adder Comparator Datapath [17] (Computation Without Feedback): To illustrate how architectural techniques can be used to compensate for reduced speeds, a simple 8-b datapath consisting of an adder and a comparator is analyzed assuming a 2.0 μm technology. As shown in Fig. 18, inputs A and B are added, and the result compared to the worst-case delay through the adder, comparator and latch is approximately 25 ns at a supply voltage of 5 V, the system in the best case can be clocked with a clock period of $T = 25$ ns. When required to run at this maximum possible throughput, it is clear that the operating voltage cannot be reduced any further since no extra delay can be tolerated, hence yielding no reduction in power. We will use this as the reference datapath for our architectural study and present power improvement numbers with respect to this reference. The power for the reference datapath is given by

$$P_{\text{ref}} = C_{\text{ref}} V_{\text{ref}}^2 f_{\text{ref}} \tag{14}$$

where C_{ref} is the total effective capacitance being switched per clock cycle. The effective capacitance was determined by averaging the energy over a sequence of input patterns with a uniform distribution. One way to maintain throughput while reducing the supply voltage is to utilize a parallel architecture. As shown in Fig. 19, two identical adder-comparator datapaths are used, allowing each unit to work at half the original rate while maintaining the original throughput. Since the speed requirements for the adder, comparator, and latch have decreased from 25 ns to 50 ns, the voltage can be dropped from 5 V to 2.9 V (the voltage at which the delay doubled, from Fig. 7). While the datapath capacitance has increased by a factor of 2, the operating frequency has correspondingly decreased by a factor of 2. Unfortunately, there is also a slight increase in the total "effective" capacitance introduced due to the extra routing, resulting in an increased capacitance by a factor of 2.15. Thus the power for the parallel datapath is given by

$$\begin{aligned} P_{\text{par}} &= C_{\text{par}} V_{\text{par}}^2 f_{\text{par}} \\ &= (2.15 C_{\text{ref}})(0.58 V_{\text{ref}})^2 \left(\frac{f_{\text{ref}}}{2}\right) \approx 0.36 P_{\text{ref}}. \end{aligned} \tag{15}$$

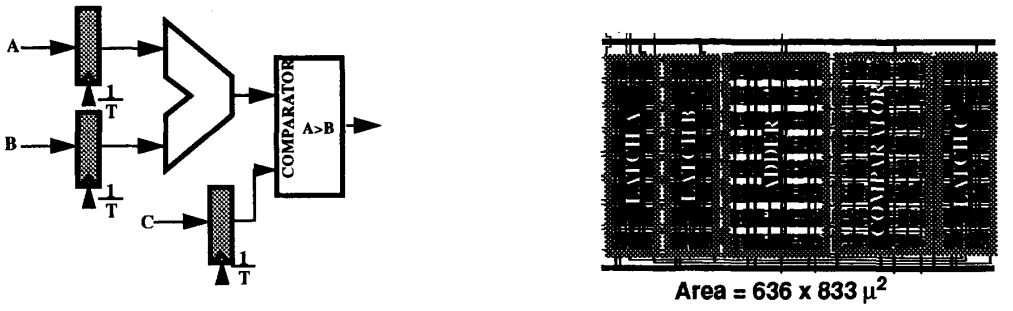


Fig. 18. A simple datapath with corresponding layout.

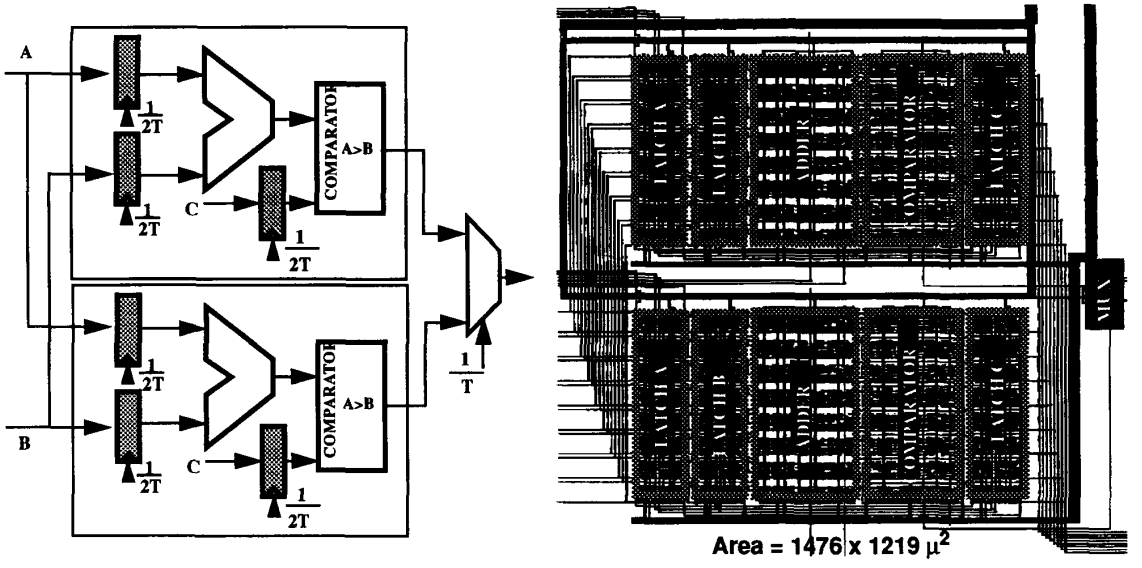


Fig. 19. Parallel implementation of the simple datapath.

The amount of parallelism can be increased to further reduce the power supply voltage and the power consumption for a fixed throughput. However, as the supply approaches the threshold voltage of the devices, the delays increase significantly with a reduction in supply voltage and therefore the amount of parallelism and corresponding overhead circuitry increase significantly. At some "optimum" voltage, the overhead circuitry due to parallelism dominates and the power starts to increase with further reduction in supply [17].

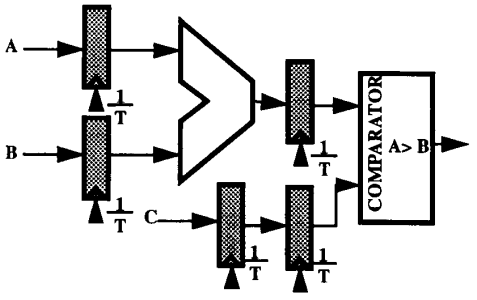
Another possible approach is to apply pipelining to the architecture, as shown in Fig. 20. With the additional pipeline latch, the critical path becomes the $\max[T_{\text{adder}}, T_{\text{comparator}}]$, allowing the adder and the comparator to operate at a slower rate. For this example, the two delays are equal, allowing the supply voltage to again be reduced from 5 V used in the reference datapath to 2.9 V (the voltage at which the delay doubles) with no loss in throughput. However, there is a much lower area overhead incurred by this technique, as we only need to add pipeline registers. Note that there is again a slight increase in hardware due to the extra latches, increasing the "effective" capacitance by approximately a factor of 1.15. The power consumed by the pipelined datapath is

$$P_{\text{pipe}} = C_{\text{pipe}} V_{\text{pipe}}^2 f_{\text{pipe}} = (1.15 C_{\text{ref}})(0.58 V_{\text{ref}})^2 f_{\text{ref}} \approx 0.39 P_{\text{ref}}. \quad (16)$$

With this architecture, the power reduces by a factor of approximately 2.5, providing approximately the same power reduction as the parallel case with the advantage of lower area overhead. As an added bonus, increasing the level of pipelining also has the effect of reducing logic depth and hence power contributed due to hazards and critical races.

Clearly an even bigger improvement can be obtained by simultaneously exploiting parallelism and pipelining. The summary of all these cases along with the area penalty is presented in Table 2.

2) Memory Access: The same parallelism concept used in the previous section can be used to optimize memory operations for low-power. For example, Fig. 21 shows two alternate schemes for reading 8 b of data from memory



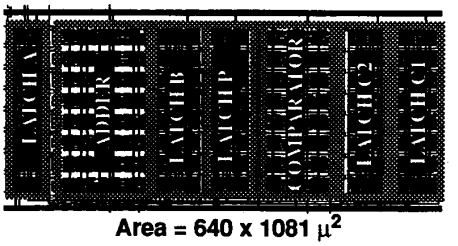


Fig. 20. Pipelined implementation of the simple datapath.

Table 2 Architecture Based Voltage Scaling Results

Architecture	Voltage	Area (normalized)	Power (normalized)
Simple	5 V	1	1
Parallel	2.9 V	3.4	0.36
Pipelined	2.9 V	1.3	0.39
Pipelined-Parallel	2.0	3.7	0.2

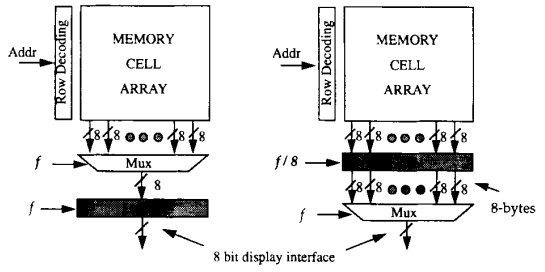


Fig. 21. Parallel memory access reduces circuit speed requirement enabling low-voltage operation.

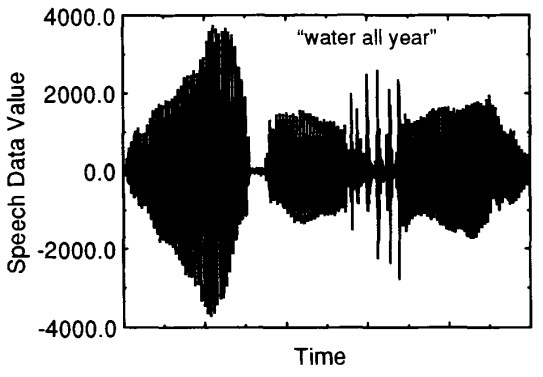
at throughput f. On the left hand side is the serial access scheme in which the 8-b of data are read in a serial format and the memory is clocked at the throughput rate f. Assume that this implementation can operate at 3 V. Another approach is to read several words from memory and clock the memory at a lower rate for the same throughput. For example, reading 8 bytes in parallel implies that the memory can be clocked at $\frac{1}{8}$ the serial rate. This implies that the time available to read the memory for the parallel implementation is 8 times as long as the serial version and therefore the supply voltage can be dropped for a fixed throughput. The parallel version can run at a supply voltage of 1.1 V (based on the delay versus V_{dd} for the SRAM presented in Section VII) while meeting throughput requirements—note that once again the multiplexor is running at the throughput rate. An important point to note is that this optimization is possible only if the data access pattern is sequential in nature, as is the case for a video frame-buffer.

B. Minimizing Switching Activity by Choice of Number Representation

In most signal processing applications, two's complement is typically chosen to represent numbers since arithmetic operations (addition and subtraction) are easy to perform. Fig. 22 shows a short segment of a speech signal and the associated transition probabilities (both the $0 \rightarrow 1$ and $1 \rightarrow 0$) assuming two's complement for each bit.

The transition probability represents the average number of transitions per cycle. There are three important regions in the transition graph: the lower-order region (the LSB's), the middle region, and the higher-order region [18]. In the lower-order region, the bits are uncorrelated both temporally and spatially and therefore the transition probability is $\frac{1}{2}$; i.e., there is a 50% probability that the bits will be in the 0 state or 1 state—a reasonable assumption for slowly varying signals. The higher-order bits represent the sign-extension operation performed in two's complement representation; i.e., the transition probability on the higher order bits is determined by the frequency at which the signal changes from positive to negative or from negative to positive. In the middle region, the transition of the bits fall between the transition probabilities of the lower-order bits and higher-order bits. The breakpoints (points of transition between the lower-order bits and the middle region and between the middle region and the higher-order bits) are determined by the mean and variance of the signal. One of the problems with two's complement representation is sign-extension, which causes the MSB sign-bits to switch when a signal transitions from positive to negative or vice-versa (for example, going from -1 to 0 will result in all of the bits toggling). Therefore using a two's complement representation can result in significant switching activity when the signals being processed switch frequently around zero and when they do not utilize the entire bit-width (i.e., the dynamic range is much smaller than the maximum possible value determined from the bit-width) since a lot of the MSB bits will perform sign extension. Even if a signal utilizes the entire bit-width, arithmetic operations such as scaling can reduce the signal dynamic range.

1) Reducing the Number of Transitions on Busses Using Sign Magnitude: One approach to minimizing the switching in the MSB's is to use a sign-magnitude representation, in which only one bit is allocated for the sign and the rest for the magnitude. In this case, if the dynamic range of a signal does not span the entire bitwidth, only one bit will toggle when the signal switches sign, as opposed to the two's complement representation where due to sign



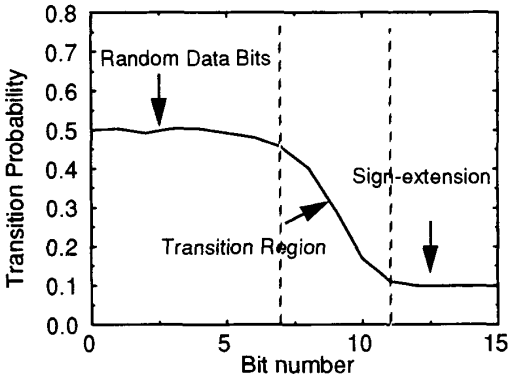


Fig. 22. Data in signal processing applications is often correlated.

extension several of the bits will switch. To illustrate this, consider gaussian data applied to a 16-b data-bus, and the let the signal have a mean of 0 and a $3\sigma = 2^{11}$. Fig. 23(a) shows the transition probabilities versus bit position number for two's complement representation as a function of the first order correlation coefficient $\rho = \text{cov}(X_n, X_{n+1})/\sigma^2$. A $\rho = 0$ indicates that the data is uncorrelated and therefore the transition probability for the MSB's is $\frac{1}{2}$ (i.e., there is a 50% chance the output is going to transition from positive to negative). A large positive correlation coefficient (e.g., +0.99) implies that the signal changes very slowly and therefore switches sign very infrequently (i.e., the transition probability is close to zero). Similarly, a negative correlation coefficient with a large magnitude (e.g., −0.99) implies that the signal changes frequently from positive to negative.

The upper breakpoint (which represents the dynamic range of signal) lies at $\log_2(3\sigma) = 11$ b in this case. Therefore for the two's complement representation, the bits from 11–15 indicate the activity of the sign bit. If a sign-magnitude representation is used (shown in Fig. 23(b)), the bits 11–14 will have a low transition activity, and bit 15, whose transitions represent the sign transition probability, will have the same activity as the two's complement representation. Also, the transition region has lower activity for the sign-magnitude representation. Clearly, there is an advantage in using sign-magnitude representation over two's complement to reduce the switching activity.

From the above example, it is clear that 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. Consider evaluating the reduction in the number of transitions as a function of dynamic range and the correlation coefficient. Let the normalized dynamic range of a signal be defined as 3σ/maximum amplitude. For a 16 b example, this turns out to be: $3\sigma\ 2^{16}-1$. Let the number of transitions represent the sum of the transitions per clock cycle for all the bits, i.e.,

$$\text{Number of Transistors} = \sum_{i=1}^{16} \alpha_i. \qquad (17)$$

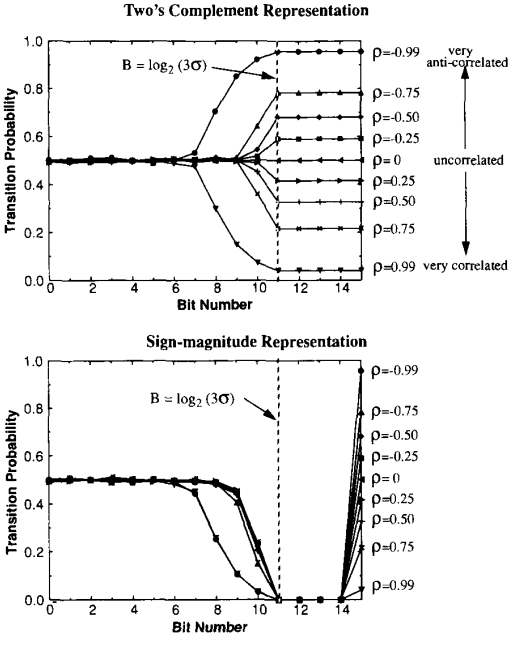


Fig. 23. Transition activity for different number representations.

Fig. 24 shows a plot of the total number of transition as a function of the normalized dynamic range and the correlation factor for two's complement representation. For a correlation factor of $\rho = 0$, the number of transitions is equal to 8 and is independent of the normalized dynamic range since the data is random and each bit has a 50% probability of transitioning ($\geq$8 transitions for a 16-b bus). For very correlated data (e.g., $\rho = 0.99$), the MSB's don't switch very often while the LSB's switch 50% of the time. Therefore, for a small normalized dynamic range, the average number of transitions is very small and for a large dynamic range it approaches the value of 8 (since the lower region with activity of 50% extends out to more bits). Similarly, for very anti-correlated data (e.g., $\rho = -0.99$), the MSB switch very frequently. Therefore, a small normalized dynamic range will result in a lot of

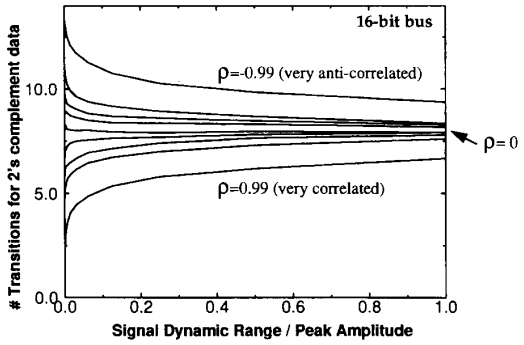


Fig. 24. Transition activity for two's complement representation.

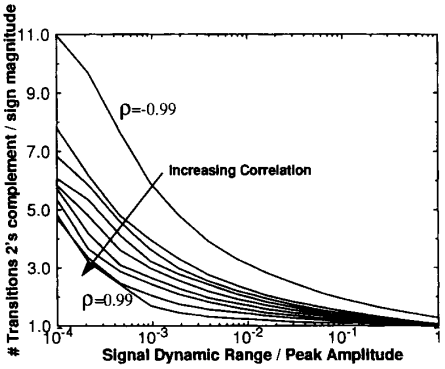


Fig. 25. Activity reduction for sign-magnitude over two's complement.

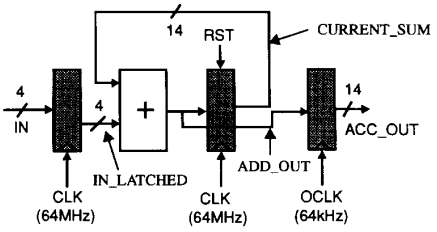


Fig. 26. Two's complement implementation of an accumulator.

transitions (since many bits are allocated to sign extension) while the number of transition approaches 8 for high values of the normalized dynamic range.

Fig. 25 shows the ratio of the number of transitions required in the two's complement representation to the sign-magnitude representation as a functions of signal correlation and normalized dynamic range (plotted on a log axis). From this plot, it is clear that the biggest win for sign-magnitude representation is when the dynamic range is smallest and the signal is very anticorrelated.

The above analysis suggests that sign-magnitude has some advantages in terms of the number of the transitions on busses. However, addition and subtraction computation are difficult to implement in sign-magnitude representation.

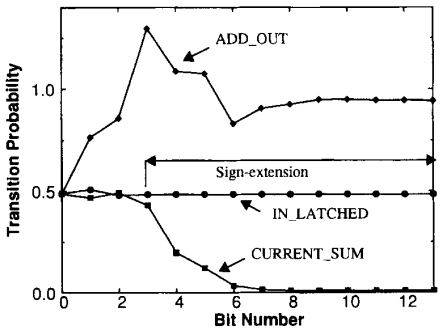


Fig. 27. Signal statistics for two's complement implementation of the accumulator datapath assuming random inputs.

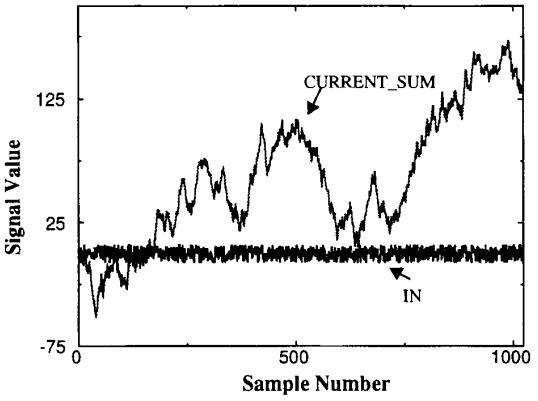


Fig. 28. Signal value for two's complement implementation assuming random inputs.

Sign-magnitude is therefore most useful for cases where large busses have to be driven (for example external memory access), where the overhead for converting back to two's complement is quite insignificant compared to the reduction in capacitance switched in the large busses. That is, a small overhead capacitance is added to the system to reduce the *overall* switched capacitance. 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, differential coding, gray-coding, etc. The coding scheme to be used is a function of signal statistics, capacitance overhead introduced, area overhead, delay, latency, etc.

2) Reducing Activity for Arithmetic Computation: As mentioned in the previous section, sign-magnitude has some advantages in terms of reducing the number of the transitions on busses, but addition/subtraction operations are difficult to implement. However, in several applications that require multiple additions inside a sample period (e.g., Multiply Accumulate Units of 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 correlator example in which the correlation length is 1024; the samples, whose values range from -7 to $+7$, are accumulated at 64 MHz. Fig. 26

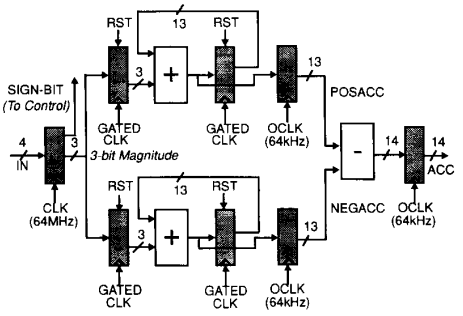


Fig. 29. Sign magnitude implementation of an accumulator.

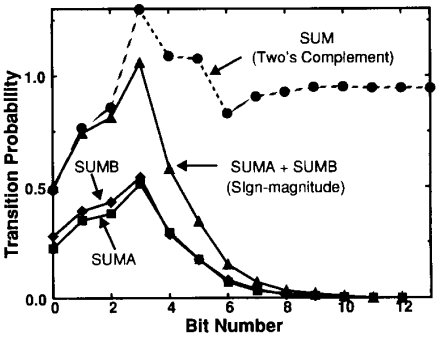


Fig. 30. Signal statistics for sign magnitude implementation of the accumulator datapath assuming random inputs.

shows a conventional architecture for an accumulator that uses two's complement representation. The accumulated result (adding 1024, 4-b numbers) is transferred to an accumulator register at 64 kHz. In this architecture, the MSB of the input, bit 3 (assuming that the LSB is bit 0), is tied to bits 4–13 of the adder input for sign-extension and therefore anytime the input switches sign, the MSB bit (which indicates the sign) will switch, resulting in all of the higher order input bits to the adder switching. Fig. 27 shows the transition activities for three signals in the two's complement datapath assuming uniformly distributed inputs. For this distribution, the input is equally likely to be positive or negative, and therefore the sign-extension bits 3–13 have a transition probability close to $\frac{1}{2}$. Since the accumulator acts as a low-pass filter (i.e., $\text{CURRENT\_SUM}_N = \text{CURRENT\_SUM}_{N-1} + \text{IN}_N$), the higher order bits have little switching activity even when the input is rapidly varying; that is, the accumulator smoothes the input signal. This is shown in Fig. 28 which shows the CURRENT_SUM output and the input value for 1024 samples (here the input is random and varying from -7 to $+7$). Although the CURRENT_SUM has low switching activity, the adder output (before the latch) has significant switching activity due to glitching, as seen from Fig. 27. The glitching activity arises since all of the input bits to the adder switch each time the input changes sign, and this results in high switching activity of the adder (even though the final adder output at the end of the cycle does not change too much relative to its value at the beginning of the cycle). Another approach for implementing the accumulator is to use a sign-magnitude representation whose datapath is shown in Fig. 29. 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 lower frequency and therefore is quite negligible in terms of capacitance overhead. The key to low-power is that there is no sign-extension is being performed and therefore the adder has a low switching activity in the higher order bits.

Table 3 Number Representation Tradeoff 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

In fact, the higher order bits only need an incrementer (as opposed to a full-adder required by the two's complement implementation), reducing the number of gates that are switched in the accumulator. Fig. 30 shows the transition activities for the output of the adders in both datapaths. Also shown in the figure is the transition activity for the two's complement implementation and the sum of the transition activities for the sign-magnitude implementation; we can see that the glitching activity is significantly reduced in the sign-magnitude implementation. 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 3 shows the power estimates 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.

C. Ordering of Input Signals

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, which is a very common operation in signal processing applications. Multiplications with constant coefficients are often optimized by decomposing the multiplication into shift-add operations and using the CSD representation. Consider the example in which a multiplication with a constant is decomposed into IN + IN $\gg$ 7 + IN $\gg$ 8. The shift operations (denoted by $\gg$)

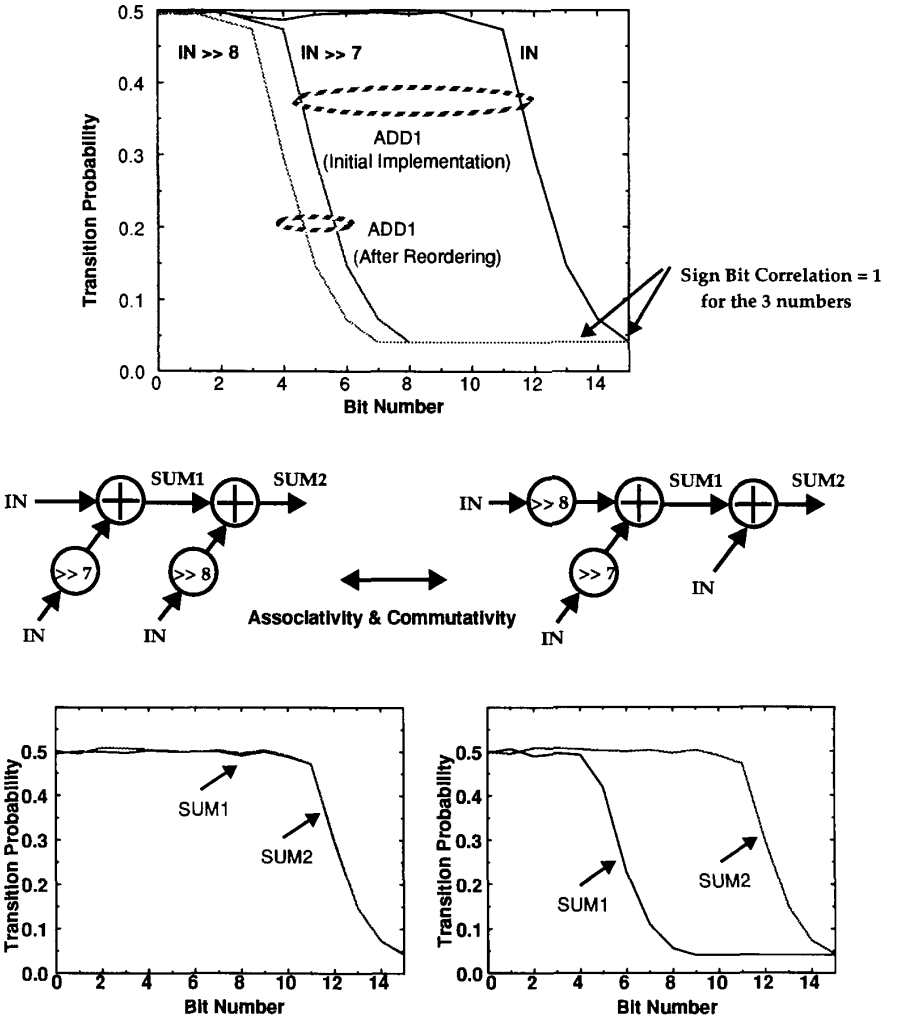


Fig. 31. Reducing activity by reordering inputs.

represent a scaling operation, which has the effect of reducing the dynamic range of the signal. This can be seen from Fig. 31 which shows the transition probability for the 3 signals IN, IN $\gg$ 7 and IN $\gg$ 8. In this example, IN has a large variance and almost occupies the entire bit-width. The shifted signal are scaled and have a much lower dynamic range. Now consider the two alternate topologies of implementing the two required additions. In the first implementation, IN and IN $\gg$ 7 are added in the first adder and the sum (SUM1) is added to IN $\gg$ 8 in the second adder. In this case, the SUM1 transition characteristics is very similar to the characteristics of the input IN since the amplitude of IN $\gg$ 7 is much smaller than IN and the 2 inputs have identical sign-bits (since a shift operation does not change the sign). Similarly SUM2 is very similar to SUM1 since SUM1 is much larger in magnitude than IN $\gg$ 8. In the second implementation (obtained by applying associativity and commutativity), the two small number IN $\gg$ 7 and IN $\gg$ 8 are summed in the first adder and the output is added to IN in the second adder. In this case, the output of the first adder has a small amplitude (since we are adding 2 scaled number of the same sign) and therefore lower switching activity. The second implementation switched 30% less capacitance than the first implementation. This example demonstrates that ordering of operation can result in reduced switching activity.

D. Minimizing Glitching Activity

As mentioned in Section II-A-4 static designs can exhibit spurious transitions due to finite propagation delays from one logic block to the next (also called critical races and dynamic hazards); that is, a node can have multiple transitions in a single clock cycle before settling to the correct logic level. To minimize the "extra" transitions and power in a design, it is important to balance all signal paths and reduce the logic depth. For example, consider the two

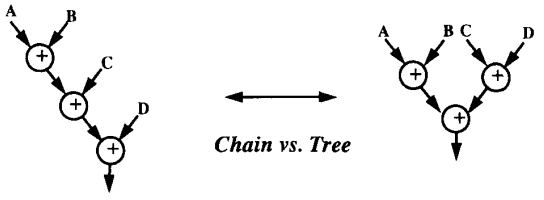


Fig. 32. Reducing the glitching activity by balancing signal paths.

implementations for adding four numbers shown in Fig. 32 (assuming a cascaded or nonpipelined implementation). Assume that all primary inputs arrive at the same time. Since there is a finite propagation delay through the first adder for the chained case, the second adder is computing with the new C input and the previous output of $A + B$. When the correct value of $A + B$ finally propagates, the second adder recomputes the sum. Similarly, the third adder computes three times per cycle. In the tree implementation, however, the signal paths are more balanced and the amount of extra transitions is reduced. The capacitance switched for a chained implementation is a factor of 1.5 larger than the tree implementation for a four input addition and 2.5 larger for an eight input addition. The results presented above indicate that increasing the logic depth (through more cascading) will increase the capacitance due to glitching while reducing the logic depth will increase register power. Hence the decision to increase or decrease logic depth is based on a tradeoff between glitching capacitance versus register capacitance. Also note that reducing the logic depth can reduce the supply voltage while keeping throughput fixed.

E. Degree of Resource Sharing

A signal processing algorithm has a required number of operations that have to be performed within the given sample period. 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 arises which is what is the architecture that 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

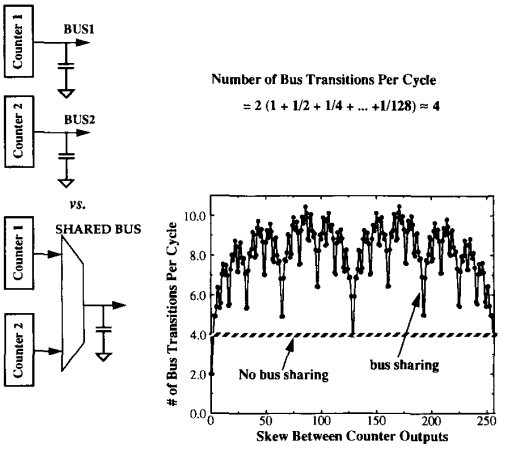


Fig. 33. Activity tradeoff for time-multiplexed hardware: Bus-sharing example.

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. However, this turns out not to be the case. To understand this tradeoff, consider two examples of resource sharing: sharing busses and sharing execution units.

1) Example 1: Time-Sharing Busses (Output of Two Counters): Consider an example of two counters whose outputs are sent over parallel and time-multiplexing busses as shown in Fig. 33. In case 1, we have two separate busses running at frequency f while case 2 has one bus running at $2f$. Therefore for a fixed voltage, the power consumption for the parallel implementation is given by

$$P_{\text{no bus-sharing}} = \tfrac{1}{2}(\sum \alpha_i) C_{nbs} V^2 f + \tfrac{1}{2}(\sum \alpha_i) C_{nbs} V^2 f = (\sum \alpha_i) C_{nbs} V^2 f \quad (18)$$

where C_{nbs} is the physical capacitance per bit of BUS1 and BUS2 (assume for simplicity that all bits have the same physical capacitance) and α_i is the transition activity for bit i (for both $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions and therefore there is a factor of $\frac{1}{2}$ in (18)). In this case, since the output is coming from a counter, the activity is $\sum \alpha_i = 1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{128}$ since the LSB switches every cycle, the 2nd LSB switches every other cycle, etc. This means that each bus will have on the average a total of approximately two transitions per clock cycle. For the time-multiplexed implementation, we have one bus C_{bs} running at twice the frequency. Typically C_{bs} will be smaller than C_{nbs} since with fewer interconnects, it is easier to route the chip and hence the wire lengths are smaller. The goal here is to show that the activity α' is modified due to time-multiplexing and the power consumption for this implementation is given by

$$P_{\text{bus-sharing}} = \tfrac{1}{2} \sum \alpha' C_{bs} V^2 2f = \sum \alpha' C_{bs} V^2 f. \quad (19)$$

Fig. 33 shows the plot of total number of transitions per cycle (i.e., $\sum \alpha'$) for the time-shared case as a function

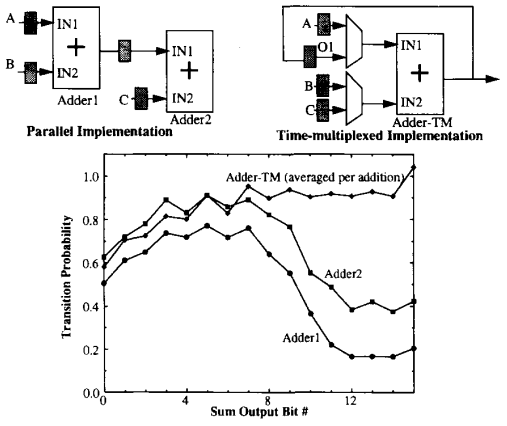


Fig. 34. Activity tradeoff for time-multiplexed hardware: Adder example.

of the skew between the counter outputs. The figure also shows the number of transitions for the nontime-shared implementation which is independent of skew and is equal to 4 (for 2 busses). 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 shows that time-sharing can significantly modify the signal characteristics and cause an increase in switching activity.

2) Example 2: Time-Sharing Execution Units: 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:

$$Y = a_0 \cdot X + a_1 \cdot X@1 + a_2 \cdot X@2 = A + B + C \quad (20)$$

where @ represents the delay operator and $A = a_0 \cdot X$, $B = a_1 \cdot X@1$ and $C = a_2 \cdot X@2$. 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 performing 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 is Fig. 34. 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 $= (N + 1) \cdot$ 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 IRSIM simulator was used to determine the switching activity for the adders in the two implementations assuming speech input data. The transition activities for the two adders for the parallel implementation and the average transition activity per addition for the time-multiplexed implementation are shown in Fig. 34. 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, IN1 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, IN2 of the time-multiplexed adder is set to B and during the second cycle, IN2 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 $X@2$ is 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).

VI. Algorithm Level Optimization

The choice of algorithm is the most highly leveraged decision in meeting the power constraints. The ability for an algorithm to be parallelized is critical and the basic complexity of the computation must be highly optimized. In this section, a few examples illustrating the tradeoffs at the algorithmic level is presented which include exploitation of concurrency and minimizing the number of operations.

A. Voltage Reduction using Algorithmic Transformations

In the Section V-A, an architecture driven voltage scaling strategy was presented in which the computation was parallelized to reduce the supply voltage and power. The examples presented in Section V-A did not have any feedback and the computation was easily parallelizable. In applications that have feedback (e.g., infinite impulse response), the computation cannot be easily parallelized and algorithmic transformations are required to alleviate the recursive bottlenecks. To illustrate the application of speedup transformations to lower power, consider a first order IIR filter, as shown in Fig. 35(a), with a critical path of 2 (assume for simplicity that each operation takes one control cycle) [19]. Due to the recursive bottleneck [20] imposed by the filter structure, it is impossible to reduce the critical path using retiming or pipelining. Also, the simple structure does not provide opportunities for the application of algebraic transformations (associativity, distributivity, etc.) and applying a single transformation is not enough to reduce power in this example. Applying loop unrolling (Fig. 35(b)) does not change the effective critical path or capacitance and therefore the supply voltage cannot be

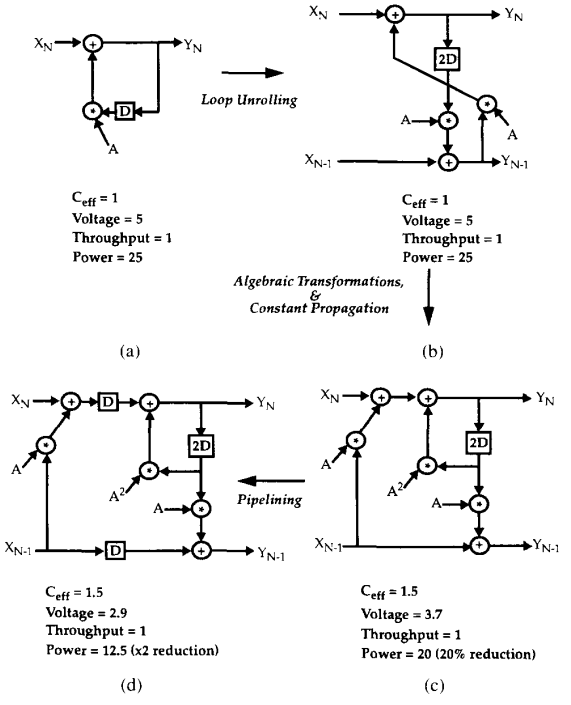


Fig. 35. Using speedup transformation to reduce power.

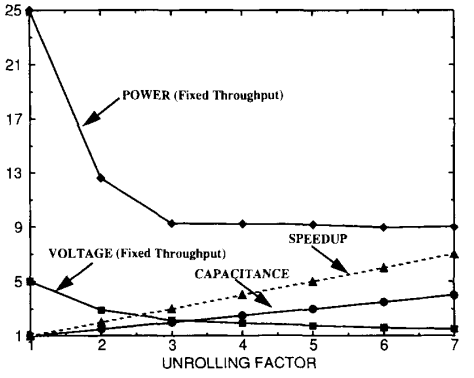


Fig. 36. Optimizing for power is different than optimizing for speed.

altered; as a result, the power cannot be reduced. However, loop unrolling enables several other transformations such as distributivity, constant propagation ($A * A = A^2$), and pipelining which result in a significant reduction in power dissipation. After applying loop unrolling, distributivity, and constant propagation in a systematic way, the output samples can be represented as

$$Y_{N-1} = X_{N-1} + A * Y_{N-2} \quad (21)$$

$$Y_N = X_N + A * X_{N-1} + A^2 * Y_{N-2}. \quad (22)$$

The transformed solution has a critical path of 3 (Fig. 35(c)). However, pipelining can now be applied to this structure, reducing the critical path further to 2 cycles (Fig. 35(d)). Since the final transformed block is working at half the original sample rate (since we are processing 2 samples in parallel), and the critical path is same as the original datapath (2 control cycles), the supply voltage can be dropped to 2.9 V (the voltage at which the delays increase by a factor of 2, see Fig. 7). However, note that the effective capacitance increases since the transformed graph requires 3 multiplications and 3 additions for processing 2 samples while the initial graph requires only one multiplication and one addition to process one sample, or effectively a 50% increase in capacitance. The reduction in supply voltage, however, more than compensates for the increase in capacitance resulting in an overall reduction of the power by a factor of 2 (due to the quadratic effect of voltage on power). This simple example can be used to illustrate that optimizing for throughput will result in a *different* solution than optimizing for power. For this example, arbitrary speedup can be achieved by continuing to apply loop unrolling combined with other transformations (algebraic, constant propagation, and pipelining). The speedup grows linearly with the unrolling factor, as shown in Fig. 36. If the goal is to minimize power consumption while keeping the throughput fixed, the speedup can be used to drop the supply voltage. Unfortunately, the capacitance grows linearly with unrolling factor (since the number of operations per input sample increases) and soon limits the gains from reducing the supply voltage. This results in an "optimum" unrolling factor for power of 3, beyond which the power consumption starts to increase again.

This example brings out two very important points: First, the application of a particular transformation can have conflicting effects on the different components of power consumption. For example, a transformation can reduce the voltage component of power (through a reduction in the critical path) while simultaneously increasing the capacitance component of power. Therefore, while speedup transformations can be used to reduce power by allowing for reduced supply voltages, the "fastest" solution is often NOT the lowest power solution. Second, the application of transformations in a combined fashion almost always results in lower power consumption than the isolated application of a single transformation. In fact, it is often necessary to apply a transformation which may temporary increase the power budget, in order to enable the application of transformations which will result in a more dramatic power reduction.

B. Minimizing the Number of Operations: Vector Quantization Example

Minimizing the number of operations to perform a given function is critical to reducing the overall switching activity. To illustrate the power tradeoffs that can be made at the algorithmic level, consider the problem of compressing a video data stream using vector quantization. Vector quantization (VQ) is a lossy compression technique which exploits the correlation that exists between neighboring samples and quantizes samples together rather than individually. Detailed description of vector quantization can be

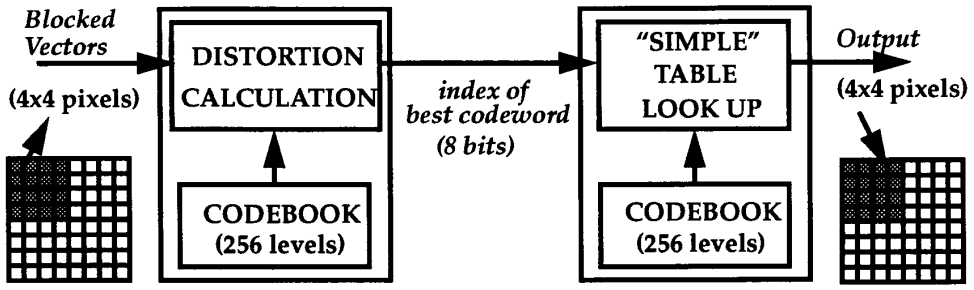


Fig. 37. Video compression/decompression using vector quantization.

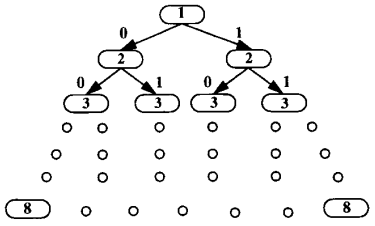


Fig. 38. Tree structured vector quantization.

found in [21]. Fig. 37 shows a block diagram of the VQ encoding/decoding process. On the encoder side, a group of pixels is blocked into a vector and compared (using a metric such as mean square error or absolute error) against a set of predetermined reproduction vectors (a set of possible pixel patterns) and the index of the best match is output. The decoder has a copy of all possible reproduction vectors (codebook) and the index of the best codeword is used to reconstruct the image using a simple table lookup operation. For this example, the image is segmented into 4 × 4 blocks (i.e., the vector size is 16 pixel words) and there are 256 levels in the codebook. In this case, 16:1 compression is achieved since only 8 b are transmitted (choosing 1 out of 256 levels) instead of 16 × 8 b for the true data. In this section, the focus will be on evaluating the computational complexity of encoding algorithms for VQ. Typically, the distortion metric used is mean square error. The distortion metric between an input vector $\mathbf{X}$ and a codebook vector $\mathbf{C}_i$ is computed as follows:

$$D_i = \sum_{j=0}^{15} (X_j - C_{ij})^2. \tag{23}$$

Three VQ encoding algorithms will be evaluated: full search, tree search, and differential codebook tree search.

1) Full Search Vector Quantization: Full search is a brute-force approach in which the distortion between the input vector and every entry in the codebook is computed. The distortion as defined in equation X is computed 256 times (for codebook entries **C0** through **C255**) and the code index that corresponds to the minimum distortion is determined and sent over to the decoder. For each distortion computation, there are 16 8-b memory accesses (to fetch the entries in the codeword), 16 subtractions, and 16 multiplications and 15 additions. In addition to this, the minimum of the 256 distortion values, which involves 255 comparison operations, must be determined.

2) Tree-Structured Vector Quantization: In order to reduce the computational complexity required by an exhaustive full-search vector quantization scheme, a binary tree-search is typically used. The basic idea is to perform a sequence of binary search instead of one large search. As a result, the computational complexity increases as $\log N$ instead of N, where N is the number of nodes at the bottom of the tree. Fig. 38 shows the structure for the tree search. At each level of the tree, the input vector is compared against two codebook entries. If for example at level 1, the input vector is closer to the left entry, then the right portion of the tree is never compared below level 2 and an index bit 0 is transmitted. This process is repeated till the leaf of the tree is reached. The TSVQ will in general have some degradation over the performance of the full search VQ due to the constraint on the search. However, this is typically not very noticeable [21] and the power reduction relative to the full search VQ is very significant. Here only $2 \times \log_2 256 = 16$ distortion calculations have to be made compared to 256 distortion calculations in the full search VQ. The number of comparison operations is also reduced to 8.

3) Differential Codebook Tree-Structure Vector Quantization: Another option is to use the same search pattern as the previous scheme but perform computational transformations to minimize the number of switching events [22]. In the above scheme, at each level of the tree, the distortion difference between the left node and right node needs to be computed. This is summarized by the following equation:

$$D_{\text{left-right}} = \sum_{j=0}^{15} (X_j - C_{\text{left}j})^2 - \sum_{j=0}^{15} (X_j - C_{\text{right}j})^2. \tag{24}$$

This equation can be manipulated to reduce the number of operations.

$$D_{\text{left-right}} = \sum_{j=0}^{15} ((Xj - C_{\text{left}j})^2 - (X_j - C_{\text{right}j})^2) \tag{25}$$

Table 4 Computational Complexity of VQ Encoding Algorithms

Algorithm	# of Memory Accesses	# of Multiplications	# of Additions	# of Subtractions
Full Search	4096	4096	3840	4096
Tree Search	256	256	240	264
Differential Search/Tree Search	136	128	128	0

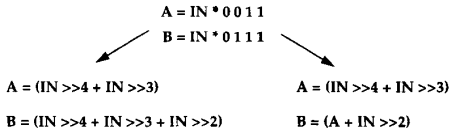


Fig. 39. Simple example demonstrating common subexpression elimination.

$$D_{\text{left-right}} = \sum_{j=0}^{15} (X_j^2 + C_{\text{left}j}^2 - 2X_jC_{\text{left}j} - X_j^2 - C_{\text{right}j}^2 + 2X_jC_{\text{right}}) \quad (26)$$

$$D_{\text{left-right}} = \sum_{j=0}^{15} (C_{\text{left}j}^2 - C_{\text{right}j}^2) + \sum_{j=0}^{15} 2X_j(C_{\text{right}j} - C_{\text{left}j}). \quad (27)$$

The first term in (27) can be precomputed for each level and stored. By storing and accessing $2 \times (C_{\text{right}j} - C_{\text{left}j})$, the number of memory access operations can be reduced; that is, by changing the contents of the codebook through computational transformations, the number of switching events—number of multiplications, additions/subtractions and memory accesses—can be reduced. Table 4 shows a summary of the computational complexity per input vector (16 pixels).

C. Minimizing Number of Operations: Multiplication with Constants

A powerful technique to reduce the number of operations is conversion of multiplications with constants into shift-add operations. Since multiplications with fixed coefficients are quite common in signal processing applications like DCT, filters, etc., the application scope of this transformation is large. The basic idea is that multiplication with 0 is a NOP and therefore a multiplication with a constant degenerates to shift-add operations corresponding to the 1's in the coefficient. Techniques and tools have been developed to scale coefficients so as to minimize the number of 1's in the coefficients so as to minimize the number of shift-add operations. In addition, if an input is being multiplied with multiple coefficients, some of the shift-add terms can be shared and the number of operations can be further reduced. Fig. 39 shows an example of exploiting multiple coefficients being multiplied with the same input. On the left is a brute force implementation in which each multiply is computed separately. On the right side is another approach which exploits common terms in the coefficients and therefore some of the shift-add terms

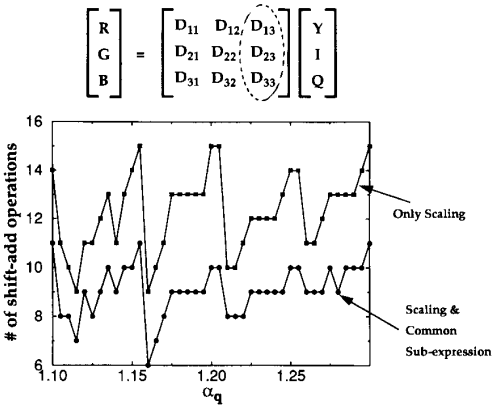


Fig. 40. Scaling and common subexpression can reduce the number of operations.

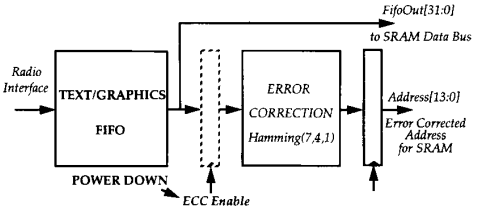


Fig. 42. Gated clocks to shut down modules when not used.

can be shared. In this implementation, A is first computed and is then used in the computation of B.

Fig. 40 shows an example of converting between the YIQ color space to the RGB color space for the video decompression module to be described later. The translation consists of a matrix multiplication with constant coefficients. The graph in Fig. 40 shows the number of shift-add operations for the three coefficients being multiplied with the Q input as a function of the scaling factor (the constant that is multiplied to the three coefficients). The top curve represents the number of operations for scaling alone and the bottom one represents scaling plus common subexpression elimination. From this figure, it is obvious that there is a great degree of freedom in optimizing the coefficients for minimizing number of operations for low-power.

VII. System Design Example: A Portable Multimedia Terminal

In this section, the various techniques presented in the previous sections will be applied to the design of a portable multimedia terminal that will allow users to have untethered access to fixed multimedia information servers [23]. The

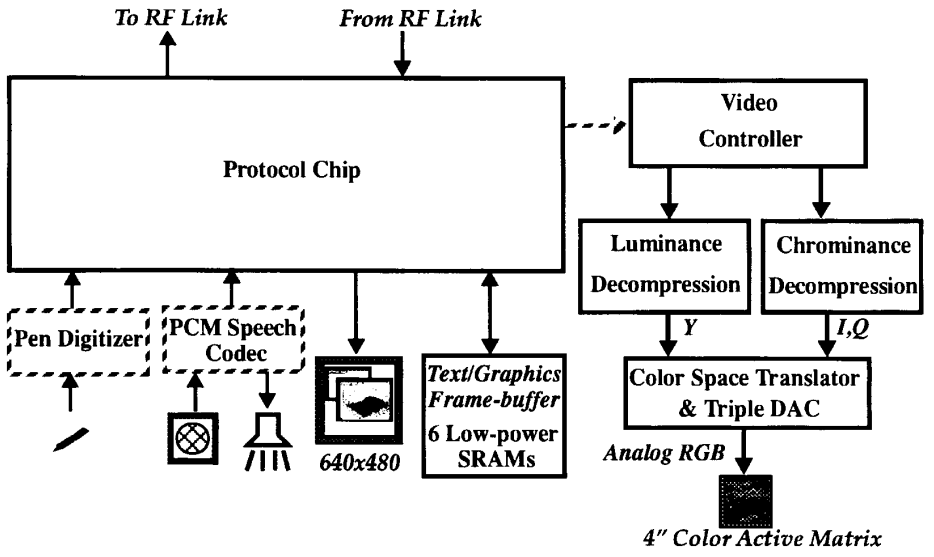


Fig. 41. Overview of the blocks in a multimedia I/O terminal.

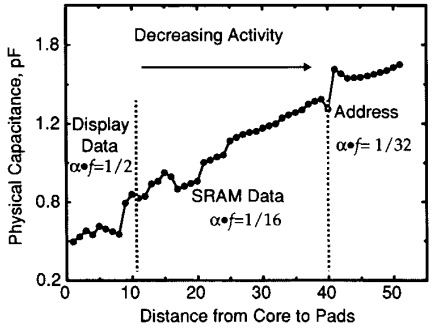


Fig. 43. Optimizing placement for low-power: Example of routing large data/control busses.

portable terminal is designed to transmit audio and pen input from the user to the network on a wireless uplink and will receive audio, text/graphics, and compressed video from the backbone on the downlink. The portability requirement resulted in the primary design focus to be on power reduction. The availability of communications between the terminal and computational resources at a fixed base station at the other end of the wireless link provides a major degree of freedom for optimizing power; i.e., any computation that does not have to be performed on the terminal (such as general purpose computation or application specific tasks such as speech and handwriting recognition) can be removed from the terminal. This resulted in a terminal that only provides the functionality to interface to I/O devices, as shown in Fig. 41. Six chips provide the interface between a high speed digital radio modem and a commercial speech codec, pen input circuitry, and LCD panels for text/graphics and full-motion video display. The chips provide protocol conversion, synchronization, error correction, packetization, buffering, video decompression, and D/A conversion at a total power consumption of less than 5 mW. A protocol chip is used to communicate between the various I/O devices in the system. On the uplink, 4 Kbps digitized pen data and 64 Kbps, μ-law speech data are buffered using FIFO's, arbitrated and multiplexed, packetized, and transmitted in a serial format to the radio modem. On the down link, serial data from the radio at a 1 Mbps rate is depacketized and demultiplexed, the header information (containing critical information such as data type and length) is error corrected and transferred through FIFO's to one of the three output processing modules: speech, text/graphics, and video decompression. The speech module communicates data serially to a codec from the FIFO at a rate of 64 Kbs. The text/graphics module corrects error sensitive information, and generates the control, timing and buffering for conversion of the 32 b wide data from the text/graphics frame-buffer (implemented using six low-power 64 Kb SRAM chips) to the 4-b at 3 MHz required by a 640 $\times$ 480, passive LCD display. The final output module is the video decompression which is realized using four chips. The algorithm used is vector quantization, which involves memory lookup operations from a codebook of 256 4 $\times$ 4 pixel patterns. Vector quantization was chosen over the JPEG algorithm to perform the video decompression since VQ requires a very low complexity decoder (and hence lower power). Compressed YIQ video is buffered using a ping-pong scheme (one for Y and one for IQ), providing an asynchronous interface to the radio modem and providing immunity against bursty errors. The amount of RAM required is reduced by a factor of 32 by storing the video in the compressed format. The YIQ decompressed data is sent to another chip which converts this data to digital RGB and then to analog form using a triple DAC which can directly drive a 4″ active matrix color LCD display. A fourth chip performs the video

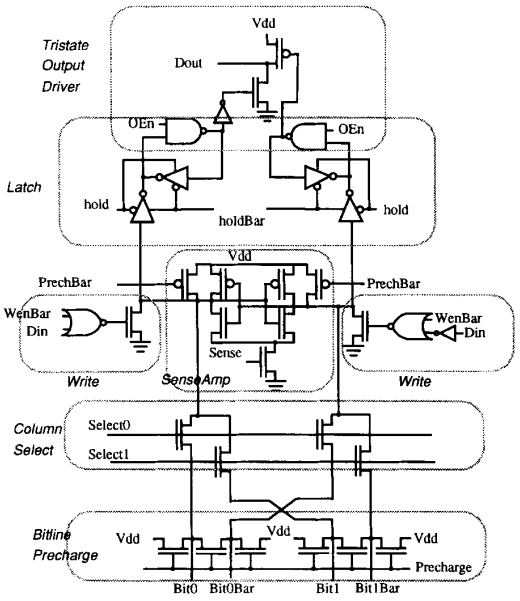


Fig. 44. Precharge, sense-amp, and "glitch" free tri-state buffer.

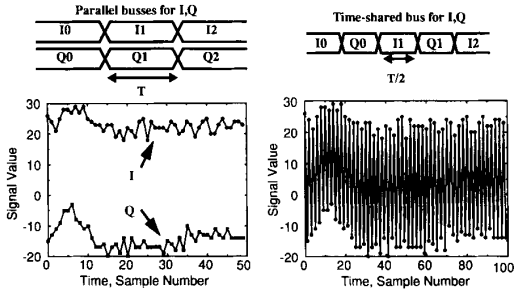


Fig. 45. Time-multiplexed hardware can increase switching activity.

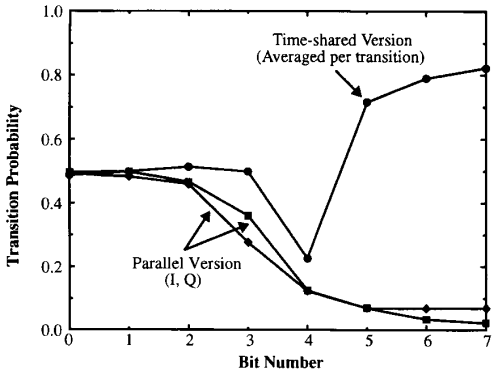


Fig. 46. Activity tradeoff for time-multiplexed versus non-time-multiplex hardware.

control functions which include the synchronization of the various chips and the LCD display, control of the ping-pong memories and loading of the codebooks, and it uses an addressing scheme which eliminates the need for an output line buffer. As described earlier, the key to low-power design is operating at the lowest supply voltage; which can result in a reduction in computational throughput. However, this can be avoided by the use of parallelism and pipelining. This approach was used extensively for both arithmetic computation and memory access, allowing the supply voltage to be as low as 1.1 V, even with a process which has $V_{tn} = 0.7$ V and $V_{tp} = -0.9$ V. To interface the low-voltage core of the custom chips to I/O devices running at 5 V, level-conversion pads are used (See Fig. 13). The entire chipset is implemented with a low-power cell library that features minimum sized transistors (as described in Section IV-B), optimized layout and logic with minimum switching activity. A few of the specific techniques used in each chip are outlined below.

A. Protocol Chip

At the logic level, gated clocks are used extensively to power down unused modules. For example, consider the error correction module for the text/graphics module. The basic protocol for the text/graphics module that is sent over the radio link and the text/graphics FIFO is address information for the frame-buffer followed by bit-mapped data. While address information is sensitive to channel errors, bit-mapped data is not very sensitive and therefore only the address information is error corrected. Fig. 42 shows a block diagram of a power efficient implementation of this function. A register is introduced at the output of the FIFO and a gated clock is used to enable the error correction module to only process the address information and the ECC is shut down during the rest of the time. In this manner, the inputs to the ECC are not switching around when the data portion of the protocol is accessed from the FIFO. Since typically, the address in only a small portion of the bandwidth compared to the data, significant power savings is possible.

At the layout level, transition activity can be used to drive the placement and routing. Fig. 43 shows an example of routing large busses from the core to the pads for the protocol chip. The text/graphics module on the protocol chip communicates to both the text/graphics frame-buffer and to the text/graphics display. The display requires 8 b (4 for the top half and 4 for the bottom half) at a rate of 3 MHz, while each SRAM block uses 32 b clocked at 3 MHz/8 (using the parallel access scheme described in Section V-A-2). The address bits are also clocked at 3 MHz/8 and have very little activity since the accesses are mostly sequential. The approach to minimize power is to route the high activity display data (and display clock) which switches every cycle to have the shortest lengths, while the SRAM address, which have an activity factor 16 times lower, are allowed to have the longest wires.

B. Frame-Buffer SRAM

The 64 kb SRAM frame buffer chips are internally organized into eight 8 kb blocks each. These memories

Table 5 Summary of Chipset for the Infopad Terminal

Chip Description	Area (mmxmm)	Minimum Supply Voltage	Power at 1.5V
Protocol	9.4 x 9.1	1.1 V	1.9 mW
Frame-buffer SRAM (with loading)	7.8 x 6.5	1.1 V	500 μW
Video Controller	6.7 x 6.4	1.1 V	150 μW
Luminance Decompression	8.5 x 6.7	1.1 V	115 μW
Chrominance Decompression	8.5 x 9.0	1.1 V	100 μW
Color Space Conversion and Triple DAC	4.1 x 4.7	1.3 V	1.1 mW

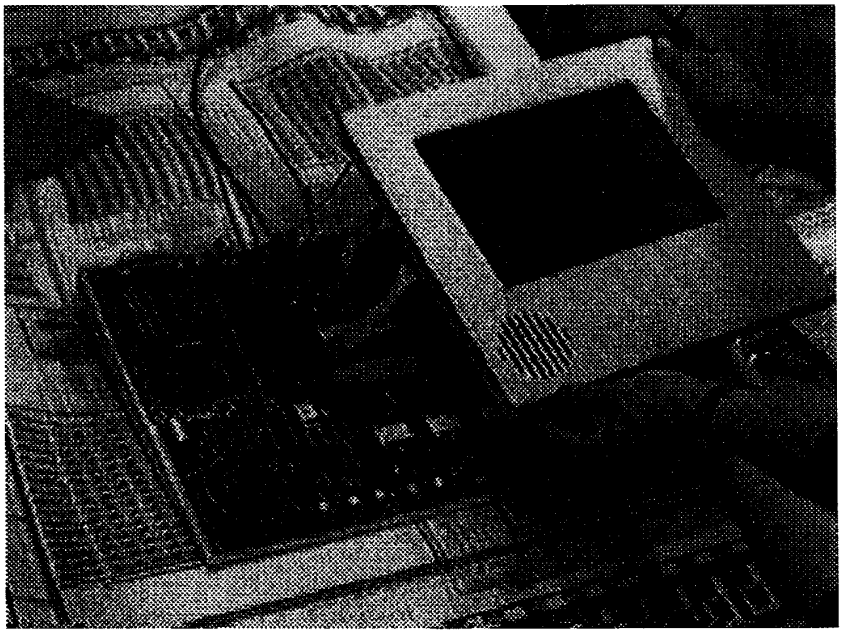


Fig. 47. Overview of the IP graphics multimedia I/O terminal.

were synthesized from a parameterized design that can be scaled on the chip, block, and subblock level making it easily retargeted for entire memory chips (as used in the frame buffer) or for on-chip memories for processors (such as in the video decompression chips). The frame buffer conserves power by minimizing both switching activity and voltage swing. At the system level, only one SRAM chip in a bank needs to be active at any given time because each chip has a full 32 b data bus. Thus by dividing the words among the chips by address (rather than by bit) only one chip consumes control overhead. The chip level architecture also minimizes switching by first decoding at the block level (using three address lines), and then activating only one of the eight memory blocks. Since each block has a 32 b data bus, minimal column decoding (in this case 2 to 1 required for sense amp pitchmatching) is needed, so only 64 bitlines are charged and discharged in the whole bank each cycle. Internally, the blocks themselves conserve power by using low-voltage-swing bitlines (as described for the FIFO in Section IV-C). The bitlines are precharged to an NMOS threshold voltage level below the supply voltage. For example, operating with a 1.1 V supply, the bitlines only swing between 0.35 V and 0.0 V.

The blocks also conserve power by eliminating glitches on the data bus. In section, it was shown that balanced paths can be used to reduce glitching for static circuits; another approach is to use self-timing as used in the SRAM design. Basically, when a block is activated to read, its output is initially tristated; only as the sense-amp reads the data is the output enabled, so there can be no spurious transitions on the data bus (Fig. 44).

C. Video Decompression and Display

The video decompression and frame-buffer are very memory intensive and the parallelism technique described in Section V-A-2 was used to access the ping-pong frame-buffers and lookup table at very low rates and run the circuits at 1.1 V. For example, the video frame-buffer was clocked at 156 kHZ while meeting the throughput rate of 2.5 MHz. In the YIQ to RGB translation, which involves multiplication with constant coefficients, the switching events are minimized at the algorithmic level by substituting multiplications with hardwired shift-add operations (in which the shift operations degenerated to wiring) and by optimally scaling coefficients. In this way, the 3 $\times$

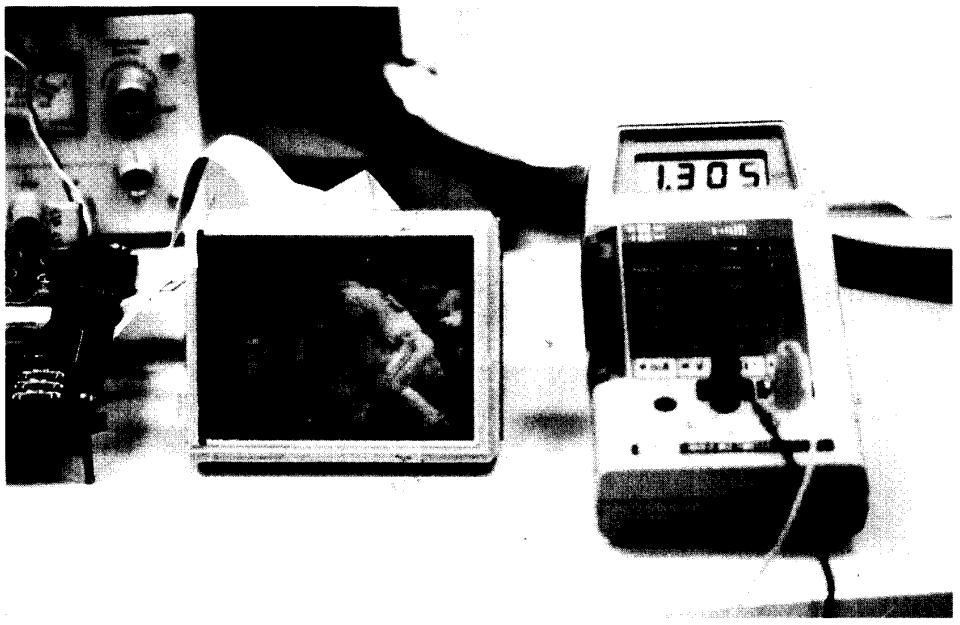


Fig. 48. Video decompression hardware output.

3 matrix conversion operation degenerated to 8 addition. The implementation was fully parallel and therefore there was no controller. For I/O communication (between the decompression chips and the color space chip) and in the matrix computation, sign-magnitude representation is chosen over two's complement to reduce the toggle activity in the sign bits. At the architecture level, time-multiplexing was avoided as it can destroy signal correlations, increasing the activity. Fig. 45 shows two alternate schemes for transmitting the I and Q data from the decompression chips to the color space converter chip. On the left is a fully parallel version in which I and Q have separate data busses. Also shown is the data for I and Q for a short segment in time. As seen, the data is slowly varying and therefore has low switching activity in the higher order bits. On the right is a time-multiplexed version in which there is a single time-shared bus in which the I and Q samples are interleaved. As seen from the signal value on the data bus, there is high switching on the data bus resulting in high activity and power. Fig. 46 shows the transition probability for both the parallel and time-multiplexed version. A low-voltage 6-b DAC based on a current switched array is used to drive the LCD. It operates down to 1.3 V, uses minimum sized devices for digital decoding, a single ended architecture to reduce the average power by a factor of 2 and a low-voltage current reference, and consumes on average 440 μW at 2.5 Mhz and 0.7 V_{pp} output amplitude when continuously operational. The actual power measured in the system was lower since the DAC was shut down during the horizontal and vertical blanking periods. The specifications of the low power chipset which was fabricated in 1.2 μm technology is given in Table 5. A PCB board containing the protocol chip, 6 SRAM chips, a speech codec, and pen interface logic has been fabricated and tested. Various power supply voltages needed for the design including –17 V (adjustable using a trim-pot) for display drive, 12 V for dc to ac inverter for the backlight, 1.1 V for the custom chips, and –5 V for the speech codec have been realized using commercial chips. This board is integrated with the Plessey radio modem to realize a complete I/O terminal with a 1 Mb/s wireless channel. Fig. 47 shows the photograph of the first generation InfoPad terminal (IPGraphics). The next version of InfoPad will provide support for one-way full motion video. Fig. 48 shows the output of the decompression chips (running at 1.3 V) on a 4″ SHARP active matrix display.

VIII. Conclusions

A variety of approaches to reduce power consumption in CMOS circuits have been presented which involve optimization at all levels of the system design. At the technology level, the key low-power consideration is threshold reduction which allows supply voltage scaling to below 1 V. An optimum threshold voltage selection based on trading leakage and switching currents was presented which indicates that the threshold voltage should be scaled to the 0.3–0.4 V range as opposed to current day technology which uses a threshold voltage on the order of 0.7 V–1 V. At next level, the physical, circuit, and logic levels are optimized for low power. This includes activity driven place and route, the use of minimum sized devices in cell design, use of reduced swing logic, and logic level optimization and power down. Optimization at the architectural level can have a major impact on power. An architecture driven voltage scaling strategy was presented in which parallel and pipelined architectures were used to scale the supply down into the 1–1.5 V range without loss in functionality.

This strategy combined with threshold reduction allows voltage reduction to about 0.5 V, which would result in two orders of magnitude of power reduction compared to conventional designs running at 5 V. Various strategies were also presented for minimizing switched capacitance at the architectural level which includes choice of number representation, exploitation of signal correlations, and minimizing glitching transitions. Finally, the choice of algorithm is the most highly leveraged decision in meeting the power constraints. The ability for an algorithm to be parallelized is critical and the basic complexity of the computation must be highly optimized. A number of the low-power techniques that were presented were applied to the design of a chipset for a portable multimedia terminal that supports pen input, speech I/O and full-motion video. The entire chipset that performs protocol conversion, synchronization, error correction, packetization, buffering, video decompression and D/A conversion operates from a 1.1 V supply and consumes less than 5 mW—which is more than three orders of magnitude lower power compared to equivalent commercial solutions.

ACKNOWLEDGMENT

The authors thank R. Allmon, T. Burd, A. Burstein, Prof. Rabaey, and A. Stratakos for their invaluable contributions.

REFERENCES

[1] N. Weste and K. Eshragian, *Principles of CMOS VLSI Design: A Systems Perspective.* Reading, MA: Addison-Wesley, 1988.

[2] H. J. M. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE J. Solid-State Circ.,* vol. SC-19, pp. 468–473, Aug. 1984.

[3] S. Sze, *Physics of Semiconductor Devices.* New York: Wiley, 1981.

[4] J. Colinge, *Silicon-on-Insulator Technology: Materials to VLSI.* Amsterdam: Kluwer, 1991.

[5] R. Swansan and J. Meindl, "Ion-implanted complementary MOS transistors in low-voltage circuits," *IEEE J. Solid-state Circ.,* vol. SC-7, pp. 146–153, Apr. 1972.

[6] T. Burd, "Low-power CMOS library design methodology," M.S. thesis, ERL Univ. Calif. Berkeley, Aug. 1994.

[7] A. Stratakos, S. Sanders, and R. W. Brodersen, "A low-voltage CMOS DC–DC converter for a portable low-powered battery-operated system," in *IEEE Power Electron. Specialists Conf.,* 1994.

[8] A. Shen, A. Ghosh, S. Devedas, and K. Keutzer, "On average power dissipation and random pattern testability of CMOS combinational logic networks," in *IEEE Int. Conf. on Computer-Aided Design,* pp. 402–407, 1992.

[9] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming sequential circuits for low power," in *IEEE Int. Conf. on Computer-Aided Design,* pp. 398–402.

[10] V. Tiwari, P. Ashar, and S. Malik, "Technology mapping for low power," in *Proc. 1993 Design Automation Conference,* pp. 74–79.

[11] C. Tsui, M. Pedram, and A. Despain, "Technology decomposition and mapping targeting low power dissipation," in *Proc. 1993 Design Automation Conf.,* pp. 68–73.

[12] M. Alidina, J. Monterio, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low power," in *1994 International Workshop in Low Power Design.*

[13] J. Yuan and C. Svensson, "High-speed CMOS circuit technique," *IEEE J. of Solid-state Circ.,* pp. 62–70, Feb. 1989.

[14] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI.* Menlo Park, CA: Addison-Wesley, 1990.

[15] M. Kakumu and M Kinugawa, "Power-supply voltage impact on circuit performance for half and lower submicrometer CMOS LSI," *IEEE Trans. Electron Devices,* vol. 37, pp. 1902–1908, Aug. 1990.

[16] D. Dahle, "Designing high performance systems to run from 3.3 V or lower sources," in *Silicon Valley Personal Computer Conf.,* pp. 685–691, 1991.

[17] A. P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circ.,* vol. 27, no. 4, pp. 473–484, Apr. 1992.

[18] P. E. Landman and J. M. Rabaey, "Power estimation for high level synthesis," in *Proc. EDAC '93,* Paris, Feb. 1993, pp. 361–366.

[19] A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Minimizing power using transformations," *IEEE Trans. Comp.-Aided Design,* Jan. 1995.

[20] K. K. Parhi, "Algorithm transformation techniques for concurrent processors," *Proc. IEEE,* vol. 77, pp. 1879–1895, Dec. 1989.

[21] A. Gersho and R. Gray, *Vector Quantization and Signal Compression.* New York: Kluwer, 1992.

[22] W. C. Fang, C. Y. Chang, and B. J. Sheu,"A systolic tree-searched vector quantizer for real-time image compression," in *VLSI Signal Processing IV.* New York: IEEE Press, 1990.

[23] A. P. Chandrakasan, A. Burstein, and R. W. Brodersen, "A low-power chipset for portable multimedia applications," in *Proc. IEEE Int. Solid State Circ. Conf.,* pp. 82–83, Feb. 1994.

Anantha P. Chandrakasan received the B.S (highest honors), M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, in 1989, 1990, and 1994, respectively.

Since 1994, he has been an Assistant Professor of Electrical Engineering at the Massachusetts Institute of Technology, Cambridge, MA. His research interests include design methodologies for low power portable systems, computer-aided design tools for VLSI, and system level integration.

Dr. Chandrakasan received an engineering fellowship from IBM in 1993. He was awarded the Analog Devices Career Development Chair at MIT.

Robert W. Brodersen (Fellow, IEEE) received B.S. degrees in electrical engineering and in mathematics at California State Polytechnic University. He received the Engineers, M.S., and Ph.D. (1972) degrees from the Massachusetts Institute of Technology, Cambridge, MA.

From 1972 to 1976, he was with the Central Research Laboratory at Texas Instruments. In 1976, he joined the Electrical Engineering and Computer Science faculty of the University of California at Berkeley, where he is now a Professor. In addition to teaching, he is involved in research involving new applications of integrated circuits, which is focused in the areas of low power design and wireless communications; and the CAD tools necessary to support this activity.

Dr. Brodersen has won best paper awards for a number of journal and conference papers in the areas of integrated circuit design, CAD and communications, including in 1979 the W. G. Baker award for the best IEEE publication in all areas. In 1983, he was corecipient of the IEEE Morris Liebmann award. In 1986 he received the Technical Achievement awards in the IEEE Circuits and Systems Society and in 1991 from the Signal Processing Society. In 1988 he was elected a member of the National Academy of Engineering.