AN APPROACH FOR POWER MINIMIZATION USING TRANSFORMATIONS

Anantha Chandrakasan[†], Miodrag Potkonjak[††], Jan Rabaey[†], Robert Brodersen[†]
[†]EECS Department, University of California at Berkeley.
[††]C & C Research Laboratories, NEC USA, Princeton.

Abstract: An approach is presented for minimizing power consumption in algorithm specific CMOS circuits using a variety of architectural and computational transformations. Several different ways are presented in which transformations can be utilized to reduce power while maintaining the throughput. In particular, the influence of specific transformations such as algebraic, control, operational and redundancy elimination are discussed. The results of applying these techniques are presented for three examples: a 11th order FIR filter, a 7th order IIR filter with a 4th order delay equalizer, and a second order Volterra filter. It is found that by applying an appropriate set of transformations an order of magnitude reduction in power can be achieved while maintaining the system throughput.

1. Introduction

Over the last few decades, most of the research and design efforts have focused on increasing the speed and throughput capabilities of digital signal processing systems. As a result, present-day technologies possess computing capacities that allow for the realization of computationally intensive tasks such as speech recognition and real-time digital video. While the focus has been on optimizing for speed and functionality, the issue of the power requirements to realize these computationally intensive signal processing functions has not been addressed. However, the growing demand for high performance "portable computing" has elevated the design for low power dissipation to be one of the most critical issues. The goal of design in these applications is to provide a low-power solution (corresponding to reduced battery requirements) without sacrificing the computational throughput necessary to perform a desired algorithm.

There are four degrees of freedom available in the design of low-power circuits and systems: technology, circuit design styles, architecture, and algorithms [1]. While each of these areas can contribute to a lower power solution, the greatest reduction comes through architecture and computation optimization which allows for a reduction of the supply voltage.

2. Sources of Power Dissipation

In CMOS technology, there are three sources of power dissipation: switching currents, short-circuit currents, and leakage currents. The switching component, however, is the only one which cannot be made negligible if proper design techniques

are followed. The power consumption due to the switching of a CMOS gate with a load capacitor, C_L, is given by the following formula [2]:

$$P_{dynamic} = p_t (C_L * V_{dd}^2 * f) \quad \textbf{(EQ 1)}$$

where f is the clock frequency, and p_t is the probability of a power consuming transition (or the activity factor). In our analysis and optimizations, we will refer to the energy per computation of a gate or module (e.g. an adder), which is given by:

$$\text{Energy per computation} = P_{total} / f_{clk} = C_{effective} V_{dd}^2 \quad \textbf{(EQ 2)}$$

where $C_{effective}$ is the average capacitance being switched per clock cycle (i.e $C_{effective} = p_t * C_{Ltotal}$).

The energy consumed by a logic block per computation is therefore a quadratic function of the operating voltage, as verified in Figure 1a, which is an experimentally derived plot of the normalized energy per computation *vs.* V_{dd}. This dependence on supply voltage has been verified for a number of logic functions and logic styles [1]. The "effective" capacitance, $C_{effective}$, for a uniformly distributed set of input values has been determined for each of the logic and memory elements in the cell library to be used in our designs. Therefore, given the voltage of operation, and the number of operations per sample period, the energy consumed by the logic elements can be estimated.

Similarly, the delays for the various logic and memory elements in the cell-library were characterized (through experimental measurement and simulations) as a function of supply voltage. Figure 1b shows a plot of experimentally derived normalized delay *vs.* V_{dd}. Once again, the delay dependence on supply voltage was verified to be relatively independent of various logic functions and logic styles [1].

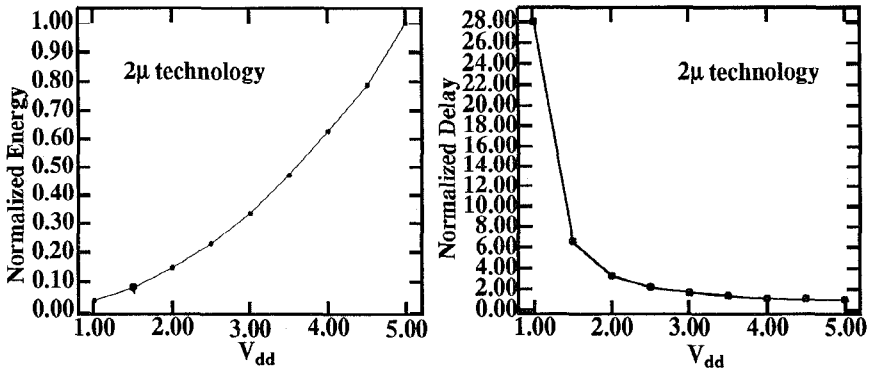

Figure 1: Plot of normalized energy vs.V_{dd} (1a) and delay vs. V_{dd} (1b).

It is clear that operating at the lowest possible voltage is most desirable, however, this comes at the cost of increased delays and thus reduced throughput. However, by modifying the architecture through a variety of transformations, the throughput can be regained, and thus a power savings can be accomplished while retaining the required functionality. It is also possible to reduce the power by choosing an architecture that minimizes the effective capacitance; through reductions in the number of operations, the interconnect capacitance, the average transition activity, the

internal bit widths and using operations that require less energy. It is these two strategies which will be pursued to minimize the power dissipation.

3. Using Transformations to Optimize Power

Transformations are changes to the computational structure in a manner that the input/output behavior is preserved. The goal is to make the new computational structure more amenable so that the number and type of used computational modules, and their interconnection are optimized. Transformations have been successfully used in several areas including optimizing compilers, logic synthesis and high level synthesis. The use of transformations makes it possible to explore a number of alternative architectures and to then choose those which result in the lowest power.

Transformations have typically been used in high level synthesis to minimize the critical path of the data-control flowgraph (i.e. maximize throughput) or to reduce silicon area while keeping the throughput constant. The approach here is to minimize a different function, namely the power dissipation of the final circuit.

3.1 Operation Reduction

The most obvious approach to power reduction, is to reduce the number of operations (and hence the number of switching events) in the data control flow graph. While this almost always has the effect of reducing the effective capacitance, the effect on critical path is case dependent. It is desirable to reduce the critical path since then the supply voltage can be reduced while keeping the throughput constant. To illustrate this trade-off, consider evaluating second and third order polynomials. Polynomial computation is very common in digital signal processing, and Horner's scheme (the final structure in our examples) is often suggested in filter design and FFT calculations when very few frequency components are needed[3]. First we will analyze the second order polynomial $X^2 + AX + B$. The left side of Figure 2a shows the straightforward implementation which requires two multiplications and two additions and has a critical path of 3 (assuming that each operation takes one control cycle). On the right side of Figure 2a, a transformed version having a different computational structure (obtained using algebraic transformations) is shown. The transformed graph has the same critical path as the initial solution and therefore the two solutions will have the same throughput at any given supply voltage. However, the transformed flowgraph has one less multiplication, and therefore has a lower capacitance and power.

Figure 2b illustrates a situation where a significant reduction in the number of operations is achieved at the expense of a longer critical path. This example, once again involves the computation of a polynomial, this time having the form X^3+AX^2+BX+C. Again, by applying algebraic transformations we can transform the computation to the Horner's scheme. The number of multiplications reduces by two, resulting in a reduction of the effective capacitance. However, the critical path is increased from 4 to 5, dictating a higher supply voltage than in the initial flowgraph for the same computational throughput. Transformations in general can have different effects on capacitance and voltage, making the associated power minimization task a difficult optimization problem.

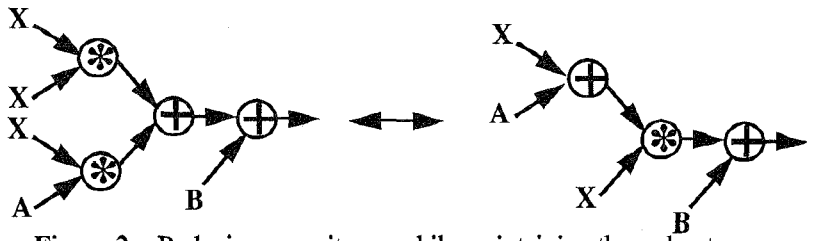

Figure 2a: Reducing capacitance while maintaining throughput.

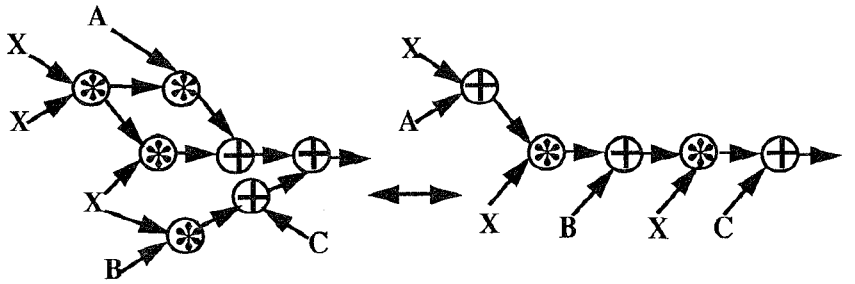

Figure 2b: Reducing capacitance at the expense of a higher supply voltage.

Transformations which directly reduce the number of operations in a data control flow graph include: common subexpression elimination, manifest expression calculation, and distributivity.

3.2 Operation Substitution

Certain operations inherently require less energy per computation than other operations. A prime example of this is strength reduction, often used in software compilers, in which multiplications are substituted for additions [4].

Another powerful transformation in this category is converting multiplications with constants into shift-add operations. Since multiplications with fixed coefficients are quite common in many signal processing applications (DCT, FFT, various types of filters, etc.), this transformation can prove to be quite beneficial.

3.3 Control Step Reduction

This is probably the single most important type of transformations for power reduction. It is not only the most common type of transformation, but often has the strongest impact on power. The goal is to reduce the number of control steps, so that slower control clock cycles and therefore lower voltages can be used. The reduction in control step requirements is most often possible due to the exploitation of concurrency. Many transformations profoundly affect the amount of concurrency in the computation. This includes loop unfolding and retiming/pipelining. We will focus on retiming in this section.

Figure 3a, shows a second order IIR filter with a critical path of 4, assuming that each addition and multiplication takes one control cycle. However, after retiming (Figure 3b), the critical path is reduced to 3 control cycles. It is easy to verify that this solution is optimum (i.e. no other retiming configuration results in a shorter critical path), since the upper left loop has three operation and only one delay and

prevents retiming from achieving a shorter critical path. This reduction of the critical path can be now used to increase the time available to each cycle, and therefore reduce the voltage and power.

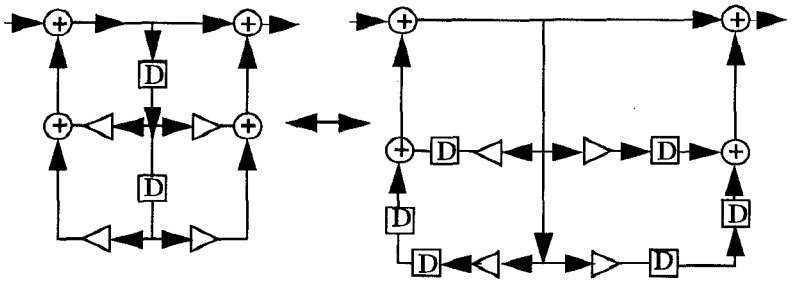

Figure 3: Retiming for reducing the critical path (4 *vs.* 3).

Since pipelining is just the generalization of retiming, the Leiserson-Saxe optimum retiming algorithm can be used for critical path minimization. Although pipelining can have some negative side effects, such as rapidly growing requirements on the number of registers used, its dramatic influence on the critical path, almost always results in significant power savings. However, in order to get the maximum improvement, it is often necessary to combine pipelining with other transformations, such as retiming for resource utilization. In the case of algebraic transformations, there are several efficient algorithms for reducing the critical path.

3.4 Resource Utilization

It is often possible to reduce the required amount of hardware, while preserving the number of control steps [5]. This is possible because after certain transformations, operations are more uniformly distributed over available time, resulting in a denser scheduling (effective utilization of the hardware). These transformations include retiming for resource utilization, associativity, distributivity and commutativity. Figure 4 shows the result of applying retiming for resource utilization on a second order IIR filter. Comparing the transformed graphs for critical path (3b) and for resource utilization (4b), we see that both solutions have a critical path of 3, however, the solution for resource utilization can be scheduled with only 2 multipliers while the solution for critical path needs 4 multipliers (since all the four multiplications can be performed only in the 3rd control step).

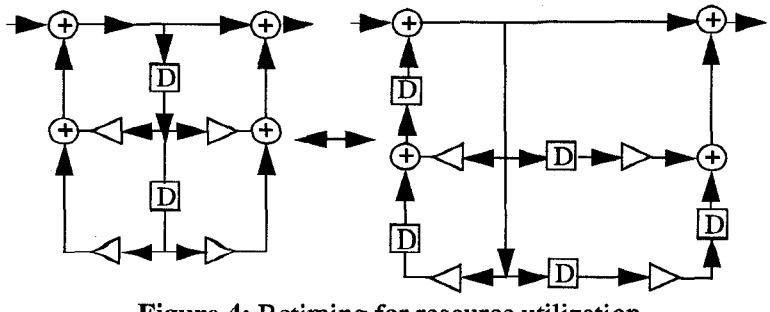

Figure 4: Retiming for resource utilization.

Smaller capacitance is achieved because there are fewer interconnects and/or fewer functional elements and registers, which are obstacles during floorplanning and routing, which indirectly influence interconnect area and capacitance. Transformations from this group use the same mechanism as transformations used for reduction of the required number of control steps. However, the goal (objective function) is different (reduced capacitance *vs.* lower voltage).

3.5 Reducing the Transition Activity

Designs using static CMOS logic can exhibit spurious transitions due to finite propagation delays from one logic block to the next. i.e. a node can have multiple transitions in a single clock cycle before settling to the correct logic value. The amount of extra transitions is a complex function of logic depth, input patterns, and skew. To minimize the "extra" transitions and power in a design, it is important to balance all signal paths and reduce logic depth. For example, consider the two implementations for adding four numbers shown in Figure 5 (assuming a chained 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 above simulations were done on layouts generated from the LagerIV [6] compiler using the IRSIM [7] switch-level simulator over 1000 random input patterns.

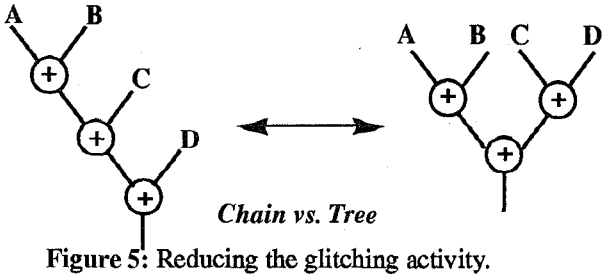

Figure 5: Reducing the glitching activity.

3.6 Wordlength Reduction

The used number of bits strongly affects all key parameters of a design, including speed, area and power. It is desirable to minimize the number of bits during power optimization for at least three reasons:

- fewer bits result in fewer switching events and therefore lower capacitance.
- fewer bits imply that the functional operations can be done faster, and therefore the voltage can be reduced while keeping the throughput constant.
- fewer bits not only reduces the number of transfer lines, but also reduces the average interconnect length and capacitance.

The influence of various transformation on numerical stability (and therefore the required wordlength) varies a lot. While some transformations, for example retiming, pipelining and commutativity, do not affect wordlength, associativity and distributivity often have a dramatic influence [8]. In some cases, it is possible to reduce both the number of power expensive operations and the required wordlength. In other cases, however, a reduction in wordlength comes at the expense of an increased number of operations.

4. Transformation Ordering

The above short survey of transformations and their influence on power optimization brings us to the conclusion that there is a large degree of freedom in choosing among the various transformations. It is important to note that often the application of a particular transformation can have conflicting effects on 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, the problem of power optimization using transformations is often a trade-off between capacitance and voltage. Also, note that applying transformations in a combined fashion results in lower power consumption compared to the isolated application of a transformation. Often, it is necessary to apply a transformation which will temporary increase the power budget, in order to enable the application of transformations which will result in a more dramatic power reduction.

To illustrate this point consider a first order IIR filter, as shown in Figure 6, with a critical path of 2. Due to the recursive bottleneck 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 and applying a single transformation is not enough to reduce power in this example. However, by applying several transformations (loop unrolling, distributivity, constant propagation, and pipelining) in a well organized fashion, a significant reduction in power dissipation can be achieved. After loop unrolling, distributivity and constant propagation, the output samples can be represented as:

$$Y_{N-1} = X_{N-1} + A * Y_{N-2} \quad \textbf{(EQ 3)}$$

$$Y_N = X_N + A * X_{N-1} + A^2 * Y_{N-2} \quad \textbf{(EQ 4)}$$

The transformed solution has a critical path of 3 (note that the delay corresponds to z^{-2}). However, pipelining can now be applied to this structure, reducing the critical path further to 2 cycles. Since the final transformed block is working at half the original sample rate (since we are processing 2 samples in parallel), while the critical path is same as the original datapath (2 control cycles), the supply voltage can be dropped to 2.9V (the voltage at which the delays increase by a factor of 2, see Figure 1b). Note that the reduction in supply voltage 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).

The results from the two previous section can be summarized in the following requirements for the application of transformations for power reduction:

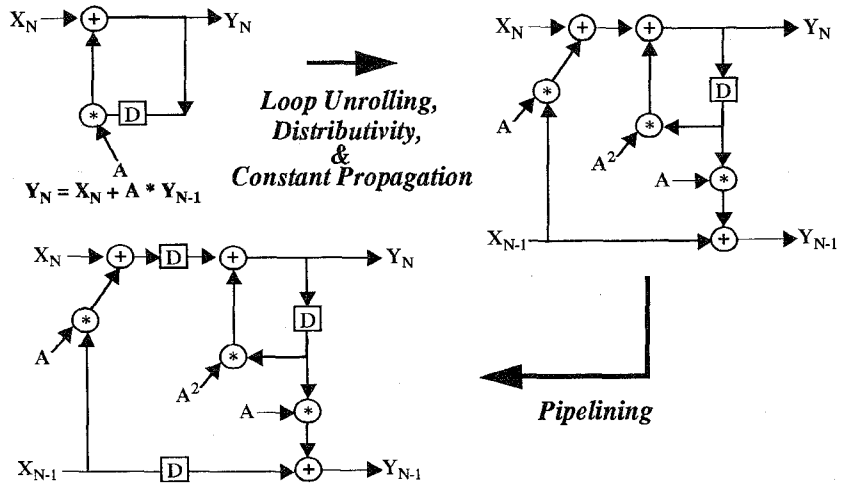

Figure 6: Ordering of transformations.

- Efficient implementation of known and new transformations so that power is an explicit part of objective function and the development of the objective function itself.
- Development of a transformation framework and search mechanism which will determine the order and extent to which the transformations are applied so that final result is globally optimal.

5. Examples and Results

The techniques described in the previous sections will now be applied to three different examples to demonstrate that a significant improvement in power can be achieved. To demonstrate the effectiveness of the proposed transformation methodology for low-power, an interactive semi-automatic approach using the HYPER high-level synthesis system was employed. Three examples of distinctly different computational structures were studied and optimized for a 2μ CMOS technology.

The first example is a 11th order FIR filter designed using the filter design program Filsyn [9]. This example is representative of a wide class of important signal processing applications such as FFT, DCT (Discrete Cosine Transform), matrix multiplication, cyclic convolution and correlation that have no feedback loops in the signal flowgraph. For this class of applications, pipelining provides an efficient and straight forward way to reduce critical paths (and thus voltage) while preserving throughput. However, for a smallest increase in area and capacitance, it is advantageous to combine pipelining with other transformations, such as retiming for resource utilization, associativity, and distributivity. For this particular FIR filter, a factor of more than 10 reduction in power was achieved through the use of various transformations (mostly pipelining) with almost no increase in area from an initial standard direct form fully time-shared realization of the filter.

The second example is a 16-bit 7th order IIR filter with a 4th order equalizer once again designed using Filsyn [9]. This example is representative of the largest class of examples where feedback is an inherent but not particularly limiting part of the

computation structure. Unlike the previous example, the FIR filter, retiming resulted in no reduction of the critical path. However, pipelining combined with retiming was very effective and allowed for the reduction of the number of control steps used from 65 to 6 cycles. The implementation area increased as the number of control cycles decreased. Figure 7a shows a plot of the number of execution units vs. the number of control cycles. The number of registers increased from an initial number of 48 (for 65 control cycles) to 102 (for 6 control cycles). This is a typical example where the area is traded off for power. It is immediately evident that the price paid for reducing critical path is the increased registers and routing overhead. Figure 7b shows a plot of the normalized clock cycle period (for a fixed throughput) as a function of number of control cycles. We see that reducing the number of control cycles through transformations allows for an increase in clock cycle period or equivalently a reduction in supply voltage. However, at very low voltages, the overhead due to routing increase at a very fast rate and the power starts to increase with further reduction in supply voltage. The power improvement was approximately 8 by dropping the voltage from 5V to approximately 1.5V.

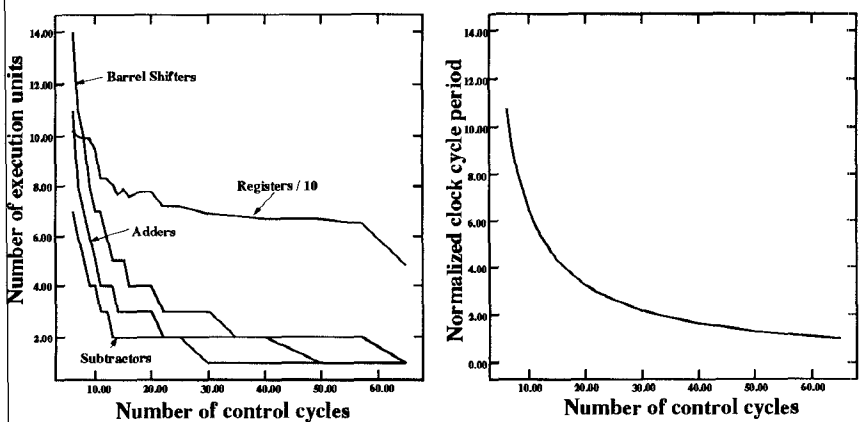

Figure 7: Plot of # of units (a) and clock cycle period (b) vs. # of control cycles for the IIR example.

The third example is a second order Volterra filter[10]. This is a particularly challenging example since pipelining cannot be applied due to a recursive bottleneck imposed by long feedback loops (optimizing with pipelining as the only transformation results in a reduction only by a factor of 1.5). By applying transformations such as associativity, distributivity and retiming, the recursive bottleneck can be alleviated allowing for a reduction in critical path. In order to achieve maximal critical path reduction for this example, it was necessary to apply loop unrolling. Figure 8 shows the layouts for optimizing the Volterra filter for area and power while keeping the throughout fixed. This example once again illustrates that area can be traded for power.

6. Conclusions

The effects of flowgraph transformations were studied for minimizing power consumption. We reviewed the various sources of power dissipation in CMOS circuits and identified several different types of transformations, which can have sig-

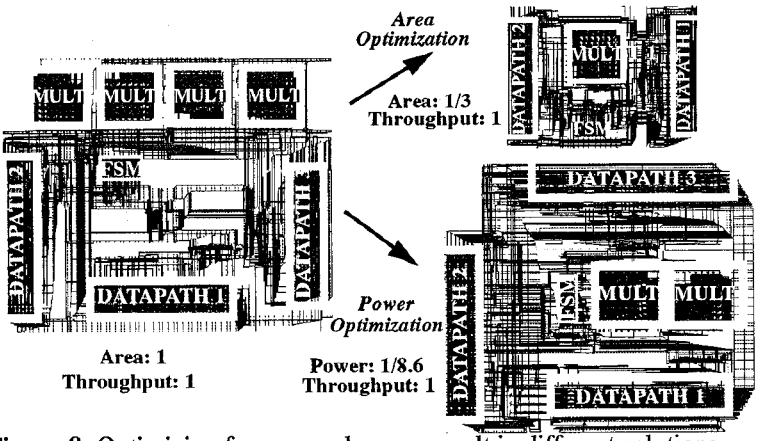

Figure 8: Optimizing for area and power result in different solutions.

nificant effect on these sources. For three different examples, of distinctly different computational structure, we applied the appropriate set of transforms for power optimization and studied power/area trade-offs for a fixed throughput. It was found that an order of magnitude improvement in power consumption could be achieved with an optimal supply voltage around 1.5V for the various examples investigated.

Acknowledgments

This research was funded by DARPA.

References

[1] A. Chandrakasan, S. Sheng, R. Brodersen, "Low-power CMOS Digital Design", IEEE Journal of Solid-state circuit, pp. 473-484, April 1992.

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

[3] G. Goertzel, "An algorithm for the Evaluation of Finite Trigonometric Series", Amer. Math. Monthly, Vol. 65, No. 1, pp. 34-35, 1968.

[4] A. V. Aho, J.D. Ullman, Principles of Compiler Design, Reading, Addison-Wesley, 1977.

[5] Rabaey, C. Chu, P. Hoang, M. Potkonjak, "Fast Prototyping of Data Path Intensive Architecture", IEEE Design and Test, pp.40-51, 1991.

[6] R. W. Brodersen, (ed.), "Anatomy of a Silicon Compiler", Klewer Academic Publishers, 1992.

[7] A. Salz, M. Horowitz, "IRSIM: An Incremental MOS Switch-level Simulator", Proceedings of the 26th ACM/IEEE Design Automation Conference, June 1989, pp. 173-178.

[8] D. Goldberg, "What every computer scientist should know about floating-point arithmetic", ACM Computing Surveys, Vol. 23, No. 1, pp. 5-48, 1991.

[9] Comsat General Integrated Systems, Super-Filsyn Users Manual, March 1982.

[10] John Mathews, "Adaptive Polynomial Filters", IEEE Signal Processing Magazine, pp. 10-26, July 1991.