Optimizing Power Using Transformations

Anantha P. Chandrakasan
Miodrag Potkonjak
Renu Mehra
Jan Rabaey
Robert W. Brodersen

EECS Department,
University of California at Berkeley

***Abstract*: The increasing demand for portable computing has elevated power consumption to be one of the most critical design parameters. A high-level synthesis system, HYPER-LP, is presented for minimizing power consumption in application specific datapath intensive CMOS circuits using a variety of architectural and computational transformations. The synthesis environment consists of high-level estimation of power consumption, a library of transformation primitives, and heuristic/probabilistic optimization search mechanisms for fast and efficient scanning of the design space. Examples with varying degree of computational complexity and structures are optimized and synthesized using the HYPER-LP system. The results indicate that more than an *order* of magnitude reduction in power can be achieved over current-day design methodologies while maintaining the system throughput; in some cases this can be accomplished while preserving or reducing the implementation area.**

1.0 Introduction

VLSI research and development efforts have focused primarily on optimizing speed to realize computationally intensive real-time tasks such as video compression and speech recognition. As a result, many systems have successfully integrated various complex signal processing modules to meet users' computation and entertainment demands. While these solutions have provided answers to the real-time problem, they have not addressed the rapidly increasing demand for portable operation. The strict limitation on power dissipation which portability imposes, must be met by the designer while still meeting ever higher computational requirements.

The desirability of portable operation of all types of electronic systems has become clear. It seems that by merely providing the same functionality of a wired system in an untethered implementation, a significant market advantage can be obtained for virtually any electronic function. Consumer electronics companies have made available portable televisions, camcoders, compact disk players, and recently hand held multimedia

devices; cordless phones provide limited mobile access for voice communications with virtually unlimited access now being provided by cellular phones; laptops, notebooks and palmtops are the fastest growing market segment of computers.

While wireless devices are rapidly making their way to the consumer market, a key design constraint for portable operation has not been considered, namely the total power consumption of the device. Reducing the total power consumption in such systems is important since it is desirable to maximize the run time with minimum requirements on size and weight allocated to batteries.

In this work, we address the problem of automatically finding computational structures that result in the lowest power consumption for a specified throughput given a high-level algorithmic specification. The basic approach is to scan the design space utilizing various flowgraph transformations, high-level power estimation, and efficient heuristic/probabilistic search mechanisms. While transformations have been successfully applied in high-level synthesis with the goal of optimizing speed and/or area, they have not addressed the problem of minimizing power.

2.0 Sources of Power Dissipation

In CMOS technology, there are three sources of power dissipation: switching, short-circuit and leakage currents. The switching component, however, is the only one that can not be made negligible if proper design techniques are followed. The switching power for a CMOS gate with a load capacitor, C_L, is given by [1]:

$$P_{switching} = \text{Energy per Transition} \bullet f = C_{avg} \bullet V_{dd}^{\ 2} \bullet f = (p_t\, C_L) \bullet V_{dd}^{\ 2} \bullet f \qquad \textbf{(EQ 1)}$$

where f is the clock frequency, and p_t is the probability of a power consuming transition (0 -> 1).

The energy consumed is therefore a quadratic function of the operating voltage, as verified in Figure 1a, which is an experimentally derived plot of the normalized energy *vs.* V_{dd}. This dependence on supply voltage has been verified for a number of logic functions and logic styles [2]. The average capacitance switched, $C_{avg} = \Sigma\, p_t\, C_L$, for a uniformly distributed set of input values has been characterized for each logic and memory element in the cell library. Similarly, the delays for the various elements in the cell-library were characterized as a function of supply voltage. Figure 1b shows experimental data of normalized delay *vs.* V_{dd} for a typical CMOS gate implemented in 2.0 μm technology. Once again, the delay dependence on supply voltage was verified to be relatively independent of various logic functions and logic styles [2].

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 at a fixed voltage: through reductions in the number of operations, the interconnect capacitance, the glitching activity, and internal bit widths and using operations that require less energy per computation.

3.0 Previous Work

In this section, a summary of the previous work done in low-power design and high-level design exploration is presented along with how this research relates to the previous efforts.

3.1 Low-power Design Techniques

The quadratic influence of voltage on power, makes voltage scaling the most attractive scheme for power reduction. A “technology” based approach proposes choosing the power supply voltage based on maintaining the speed performance for a given submicron technology [3]. 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. This was found to achieve a 60% reduction in power for a 3.3V system when compared to a 5 volt operation [4]. [2] presents an architecture based voltage scaling strategy that results in an optimal voltage for power that is much lower (in the 1-1.5V range) than obtained from the technology based scaling. The idea is to maintain throughput at reduced supply voltages through hardware duplication or pipelining. By using parallel, identical units, the speed requirements on each unit are reduced, allowing for a reduction in voltage. This work presents an efficient transformation based high-level synthesis approach to explore the architecture based voltage scaling strategy [2].

Power reduction can also be achieved by minimizing the capacitance switched at all levels of the design. There are a number of options available in choosing and optimizing the basic circuit approach and topology for implementing various logic and arithmetic functions. A pass-transistor logic family was found to minimize physical capacitance when compared to a conventional CMOS logic family [5]. At a another level, there are various topological choices for implementing a given function. For example, an adder can implemented using ripple-carry or carry-lookahead approaches. The power trade-off between various types of adders and multipliers were investigated in [6] and they concluded that a carry-lookahead topology was the “best” after taking into

account the speed-capacitance trade-off. Optimizing transistor sizing is yet another degree of freedom available in minimizing the energy. To minimize the parasitic capacitances, it is desirable to use minimal sized devices as much as possible, which also assists in minimizing the interconnect routing lengths and their associated parasitic capacitances [2]. To minimize the total switched capacitance in random logic modules, several logic synthesis optimization techniques have been proposed to lower the power dissipation [7].

3.2 Transformations in High-level Synthesis

Over the last few years, several high-level synthesis systems have incorporated comprehensive sets of transformations, coupled with powerful optimization strategies. Example systems with elaborate applications of transformations are Flamel [8], SAW [9], SPAID [10], HYPER [11], and CATHEDRAL [12].

Among the set of transformations used by the Flamel design system are loop transformations, height reduction and constant propagation. SAW uses among other transformations in-line expansion, dead code elimination, four types of transformations for conditional statements and pipelining as supporting steps during the behavioral and structural partitioning. The SPAID system's set of transformations includes retiming and pipelining, interleaving, substitution of multiplications with constants by addition and shifts and algebraic transformations. HYPER uses more than 20 different transformations whose application is supported by several probabilistic optimization algorithms. CATHEDRAL emphasizes on support for loop transformations. All systems provide interactive frameworks where the designer explores the influence of transformation mechanism or optimization algorithms for a specific transformation.

The systems described above use transformations to optimize design parameters such as area and throughput. This work addresses the problem of how high-level flowgraph transformations can be used to reduce the power consumption of VLSI implementation.

3.3 Power Estimation

Most of the work done in power estimation falls under three categories: gate-level probabilistic estimation, switch-level estimation, and circuit-level estimation. The primary trade-off between these approaches is the computational complexity *vs.* accuracy. Circuit-level approaches result in the most accurate estimates, while being the most computationally intensive.

Gate-level probabilistic approaches estimate the internal node activities of a network given the distribution of the input signals [13], [14], [15]. Once the signal probability for each node in the network is determined, the total average capacitance switched is then estimated as $\sum p_{i0} (1-p_{i0}) C_i$, where p_{i0} is the probability that node i

will be in the *ZERO* state, and C_i is the physical capacitance associated with node i. The total power is then estimated as $C_{avg} * V_{dd}^2 * f_{clk}$

An approach for estimating the power consumption in CMOS circuits using a switch-level simulator is presented in [16]. The basic idea is to monitor the number of times each node in the circuit transitions during the simulation period. C_{avg} is given by $\sum N_i / N C_i$, where N_i is the total number of power consuming transitions for node i, N is the number of simulation cycles, and C_i is the physical capacitance of node i. This approach (using the IRSIM simulator [17]) was used to estimate power at the layout level. The results from a few fabricated chips, indicate that the predicted power from IRSIM is within 30% of the measured power.

At the lowest level, power can be estimated using a circuit simulator such as SPICE. The design is described at a very low-level and the simulator accounts for short-circuit and leakage components of power. The major problem with this approach is the long simulation time which limits the number of patterns that can be applied.

The approaches mentioned above estimate power consumption from a low-level of abstraction. To use these approaches in a high-level synthesis framework, the high-level representation of the algorithm has to be mapped to a low-level description (gate or transistor level), which is very time consuming. Also, the estimation time for each new topology is too long to meaningfully explore many architectures. Hence, power must be estimated efficiently from a high level of abstraction. Recently, techniques to estimate the power consumption at the architecture level (after the flowgraph has been scheduled) have been developed [18]. In this work, power is estimated from an algorithmic level so the design space can be quickly explored.

4.0 Using Transformations to Optimize Power

Transformations are changes to the computational structure in a manner that the input/output behavior is preserved. The number and type of computational modules, their interconnection and their sequencing of operation are optimized. The use of transformations makes it possible to explore a number of alternative architectures and to choose those which result in the lowest power. We will use the control-data flow graph format to represent computation [11], in which nodes represent operations, and edges represent data and control dependencies. Transformations have primarily been used until now to optimize either the implementation area or the system throughput. The goal here is to optimize a different function, namely the power dissipation of the final circuit while meeting the functional throughput of the system. Two key approaches are used to reduce power for a fixed throughput: reducing the supply voltage by utilizing speed-up transformations (section 4.1) and reducing the effective capacitance being switched using a variety of transformations (sections 4.2 - 4.6).

4.1 Control Step Reduction to Lower Supply Voltage

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 basic idea is to reduce the number of control steps, so that slower control clock cycles can be used for a fixed throughput, allowing for a reduction in supply voltage [2]. 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 retiming/pipelining, algebraic transformations and loop transformations.

To illustrate the application of speed-up transformations to lower power, consider a first order IIR filter, as shown in Figure 2a, with a critical path of 2. Due to the recursive bottleneck [19] 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. Applying loop unrolling (Figure 2b) does not change the effective critical path or capacitance and therefore the supply voltage cannot be altered; as a result, the power cannot be reduced. However, loop unrolling enables several other transformations (distributivity, constant propagation, 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:

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

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

The transformed solution has a critical path of 3 (Figure 2c). However, pipelining can now be applied to this structure, reducing the critical path further to 2 cycles (Figure 2d). 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 2.9V (the voltage at which the delays increase by a factor of 2, see Figure 1b). 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 used to illustrate that optimizing for throughput will result in ***different*** solution that 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 Figure 3. 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 bring 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 speed-up transformations can be used to reduce power by allowing for reduced supply voltages, the “fastest” solution is often NOT the lowest power solution. The design environment described later can be used to automatically explore the design space using speed-up transformations (as described above) and find the “optimum” solution which trade’s off capacitance and voltage for a fixed throughput. 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 temporarily increase the power budget, in order to enable the application of transformations which will result in a more dramatic power reduction.

A simple first order IIR filter was presented above to demonstrate that the optimization for throughput is very different than optimization for power. Now consider a bigger and more widely used example, the DCT (Discrete Cosine Transform), to illustrate this point further. We will compare two implementation of the DCT: one proposed by Feig and Winograd [20] and the other, the direct maximally fast form. Feig’s DCT algorithm can be derived from the direct form using the exceptionally sophisticated application of common subexpression elimination/replication and algebraic rules. The direct form can be derived from the Feig’s DCT, by the simple application of the transformation set using the procedure for maximally fast implementation of linear computation [21]. While Feig’s DCT has a critical path of 11 cycles, the maximally fast DCT has a critical path of only 7 cycles. Therefore the Vdd can be reduced from 5 V to 3.25 V. While the reduction of the supply was relatively significant, the effective capacitance increased at a rate larger than an order of magnitude. Feig’s form has 54 multiplications and 462 additions while the maximally fast form has 4,096 multiplication and 4,096 additions. The power increases by a factor of more than 20, even after taking into account the voltage reduction due to throughput improvement.

The examples presented in this subsection clearly indicate that there is a sharp difference in the objective function for throughput and power. A number of key transformation for throughput enhancement have the

effect of increasing the number of operations or increasing the interconnect area. For example, it has been shown that common subexpression can yield, for many classes of design implementation, arbitrarily high throughput at the expense of additional number of operations [21]. Similarly, it has been demonstrated that retiming for throughput often results in designs with exceptionally high interconnect requirements [22].

4.2 Operation Reduction

The most obvious approach to reduce the switched capacitance, is to reduce the number of operations (and hence the number of switching events) in the data control flow graph. While reducing the operation count typically has the effect of reducing the effective capacitance, the effect on critical path is case dependent. To illustrate this trade-off, consider evaluating second and third order polynomials. Computation of polynomials 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 [23].

First we will analyze the second order polynomial $X^2 + AX + B$. The left side of Figure 4a 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 4a, 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 4b 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 of 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. Once again, this example shows that a transformation can have different effects on capacitance and voltage, making the associated power minimization task a difficult optimization problem.

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

4.3 Operation Substitution

Certain operations inherently require less energy per computation than other operations. A prime example of transformation which explores this trade-off is strength reduction, often used in software compilers, in which multiplications are substituted by additions [24]. Although this situation is not as common as the ones presented in the previous section, sometimes it is possible to achieve significant savings using this type of trade-off.

Unfortunately, this type of transformation often comes at the expense of an increase in the critical path length. This point is illustrated in Figure 5, which shows the frequently used application of redundancy manipulation, distributivity and common subexpression in complex number multiplication. A_i and A_r are constants. While the second implementation has a lower effective capacitance, it has a longer critical path. Fixing the available time to be a constant, we see that the second implementation requires at least three control cycles (assuming each operation take one control cycle), while the first one requires only two. This implies that the voltage in the first implementation can be dropped lower than the voltage in the second implementation, since the same computational throughput can be met with a 50% slower clock rate.

A powerful transformation in this category 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. Table 1 shows the power breakdown for a 11 tap FIR filter before and after this transformation. The power consumed by the execution units is reduced to one-eighth of the original power and the power consumed in the registers is also reduced. A small penalty is paid in the controller but there is a gain in the interconnect power due to a reduction in the length of the bus lines (since the final implementation has a smaller area), resulting in a total power savings of 62%.

4.4 Resource Utilization

It is often possible to reduce the required amount of hardware, while preserving the number of control steps [22]. This is possible because after certain transformations, operations are more uniformly distributed over the available time, resulting in a denser schedule. These transformations include retiming for resource utilization, associativity, distributivity and commutativity. Figure 6 shows the result of applying retiming on a second order IIR filter. Both the transformed graphs, 1 and 2, are obtained from retiming and have a critical path of 3; however, the transformed graph 2 can be scheduled with only 2 multipliers while the first needs 4 multipliers (since all the four multiplications can be performed only in the 3rd control step).

Given that there is a degree of freedom in choosing the amount of resources used for a fixed throughput (i.e at a fixed supply voltage), the question then becomes is the solution with the minimal amount of resources, as chosen for the minimal area implementation, the “best” for low-power. On one hand, reducing the amount of resources can reduce the wiring capacitance since there are fewer interconnects and/or fewer functional elements and registers, which are obstacles during floorplanning and routing. However, the amount of multiplexors and control logic circuitry will typically increase with more time-sharing of resources. Therefore the optimization strategy (and hence the power estimation model) must take into account the trade-off between interconnect capacitance and control circuitry (which determines the effective capacitance being switched).

4.5 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 (called critical races or dynamic hazards). 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, statistics of the input patterns, and skew. 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 implementations for adding four numbers shown in Figure 7 (assuming a cascaded or non-pipelined 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 by the LagerIV silicon compiler [25] using the IRSIM [17] switch-level simulator over 1000 uncorrelated random input patterns.

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 trade-off between glitching capacitance vs. register capacitance. Also note that reducing the logic depth can reduce the supply voltage while keeping throughput fixed.

4.6 Wordlength Reduction

The number of bits used 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 reduce the number of transfer lines, but also reduce 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 [26]. 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.

To illustrate the importance of wordlength optimization, consider the direct form and parallel form implementations of an 8th order Avenhaus bandpass filter (as shown in Figure 8a and 8b). The critical path of the direct form, after multiplication to shift and addition substitution is 20 clock cycles while the critical path of the parallel form is 28 cycles, assuming that each operation, in both cases, takes one cycle. However, the numerical stability of the parallel form is significantly higher, resulting in wordlength requirements of only 11 bits, while the direct form solution requires 23 bits. The effective critical path for the direct form solution is 980 ns, while parallel form has a critical path of only 610 ns. Therefore, taking wordlength requirements into consideration, changes the situation from one where the critical path in the parallel form is almost 50% longer, to one where the critical path in the direct form is more than 50% longer. Similarly, a detailed analysis of these two alternatives shows that the power consumption of the final implementation of the parallel filter is reduced by a factor of four for a given throughput. Note that the direct form can be transformed to the parallel form (or vice-versa) using a specific ordered set of transformations [27].

The results from section 4 can be summarized in the following requirements for the application of transformations for power reduction:

- Efficient implementation of known and new transformations so that power is an explicit part of the cost function and the development of the cost 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.

5.0 Cost Function

The goal is to develop an objective function that is highly correlated to the final (and unknown) power dissipation of the circuit. The objective function should be very easy to compute since it has to be evaluated many times during the optimization process. A detailed estimation, while being accurate, will require hardware map-

ping and compilation steps to convert a flowgraph to layout, making it impractical during the optimization process. Hence a model correlated to the power must be developed strictly from the flowgraph level. This involves a statistical study of the effects of various high level parameters on interconnect capacitance, control capacitance, etc.

The power consumption of a circuit (represented in a control data flowgraph) is given by:

$$P_{total} = C_{total} * V^2 * f_{sampling} \quad \textbf{(EQ 4)}$$

The goal of power optimization in this work is to keep the throughput constant (i.e. $f_{sampling}$ is fixed) by allowing the supply voltage to vary. For a fixed sample period, power optimization is equivalent to minimizing the total energy switched, $C_{total}\, V^2$, where V is appropriate voltage required to meet the throughput rate.

Figure 9 shows the hardware model used in HYPER. A datapath module contains execution units, register files, multiplexors, and buffers. Data stored in the register files is read out to the execution unit whose result is written back into register files in the same datapath or in other datapaths. The register file may be preceded by multiplexors depending on whether the execution unit receives data from different sources at different control cycles in the sample period. The execution units on different datapaths communicate through global busses and the outputs of the execution units are tri-stated if two different units share a common global bus. A distributed control approach (which contains a global and local controller) is used to sequence through the operations in the control cycles. This section addresses the estimation of the capacitance components and the supply voltage.

5.1 Capacitance estimate

The total capacitance switched depends on four components:

$$C_{total} = C_{exu} + C_{registers} + C_{interconnect} + C_{control} \quad \textbf{(EQ 5)}$$

The capacitance estimation is built on top of an existing estimation routine in HYPER that determines bounds and activity of various execution, register and interconnect components as well as the implementation area [28], [29]. The details of the capacitance estimation routines are described below.

5.1.1 Execution Units

The capacitance switched by the execution units is estimated by multiplying (over all types of operations) the number of times the operation is performed per sample period and the average capacitance per access of the operation. The total capacitance is hence given by:

$$C_{exu} = \sum_{i=1}^{numtypes} N_i \cdot C_i \quad \text{(EQ 6)}$$

where *numtypes* is the total number of operation types, N_i the number the times the operation of type *i* is performed per sample period (or the activity), and C_i is average capacitance per execution of operation type *i*. The average capacitance per execution has been characterized for the various modules (through SPICE and IRSIM simulations, using models that were calibrated with results from experimental measurements) for a uniformly distributed set of inputs. In general the probabilities are not uniform, however, this assumption is made to simplify the cost evaluation. The capacitance values are parameterized as a function of bit-width and are accessed when computing the power contributed due to the execution units. For example, the capacitance per access for a ripple carry adder as a function of the Bitwidth (using a linear least squares fit) is shown in Figure 10. Table 2 shows the high-level capacitance models for various modules in the hardware library.

The model presented above assumes the capacitance contribution due the execution units is relatively independent of allocation since the required number of operations must be performed within the sample period. For example, if a flowgraph has five additions, it is assumed that it does not matter whether the final implementation has one adder performing all the five additions within the sample period or has five adders each performing one addition within the sample period. This is not completely correct since the capacitance switched is a function of signal correlation. Techniques to model signal correlation at a high-level have been recently developed [18].

5.1.2 Registers

Registers are treated the same way as the execution units. The existing estimation program gives information about the total number of register accesses (read/write) within a given sampling period. This is essentially the “activity” of the registers. For the purposes of calculating the register energy, it is enough to know this “activity” and the actual number of physical registers is not required. The number of register accesses is multiplied with the average capacitance per register access to yield a register contribution given by:

$$C_{register} = N_{registers} \cdot C_{registers} \quad \text{(EQ 7)}$$

While the total number of registers is not important in calculating the register switching capacitance, it will affect floorplanning and chip area and therefore the interconnect capacitance. Gated clocks are used and the clock capacitance is taken into account during the characterization of the registers. Each register has a control slice that locally buffers the incoming clock.

5.1.3 Interconnect

While estimating the power consumed by the execution units and registers is quite simple and accurate, estimating the interconnect component is a very difficult and challenging task. Driven by yield, floorplanning and synthesis considerations for throughput and area optimization, several elaborate prediction models for total chip and interconnect area have been built and successfully used [30], [31]. However, high-level synthesis adds additional requirements on the prediction tools next to accuracy; during the optimization process in high level synthesis, it is necessary to estimate the final cost frequently and therefore computationally intensive models are prohibited, regardless of their precision.

Estimating the interconnect component of the final implementation should not only take into account the effects of a wide spectrum of high level synthesis tools, such as assignment, allocation, scheduling and partitioning into macro blocks, but also the effects of many low-level CAD tools, such as placement, floorplanning and global and detailed routing. Of course, an accurate model for such a complex system can be built only when a particular set of design tools is targeted. As mentioned earlier, we targeted the HYPER high-level synthesis tools and the Lager IV silicon assembler [25]. We used the scalable CMOS design rules provided by Mosis and targeted feature sizes of 1.2μm and 2μm.

The selection of this particular suit of design tools, enabled us to somewhat simplify the estimation process. We concentrated our attention only on the inter-block (between macro blocks, e.g. different datapaths) routing capacitance. The effect of intra-block (between logic modules inside a datapath) routing capacitance is already taken into account during the calculation of execution units and register contribution, by incorporating an average loading capacitance (determined for the datapath compiler used in the LagerIV silicon assembler).

Building a model which will take into account the effects of the various tools mentioned above is a formidable task. An extensive experimental study, followed by in-depth statistical analysis and verification is the only viable solution which will satisfy the contradictory requirements of modeling a complex system, with high accuracy in a computationally efficient manner.

The model for interconnect capacitance was built using fifty examples which were mapped from their Silage descriptions to layout using the HYPER synthesis system and the LagerIV silicon assembler. The selected examples cover a wide variety of DSP applications including linear and nonlinear filters (including FIR, direct form, cascade, parallel, continuous function, ladder, wave digital, Rao-Kailath IIR, polynomial and homomorphic filters), fast transformations algorithms (FFT and DCT), several video and image processing algorithms and audio examples. Selecting the set of examples for building the model was guided by the goal of including

as diverse and as typical examples as possible. While the smallest example had only 12 operations, the largest one had more than 400 operations. One half of examples was pipelined to various extents, and a subset of the rest were transformed using several transformation in different orders. The examples cover a wide variety in the ratio of critical path to available time, amount of parallelism, types of used operations, level of multiplexing, size of the final implementations and other parameters.

After assembling the results for half the examples (for 25 examples), we started building the statistical model. It immediately became apparent that the best correlation is one between the total interconnect capacitance and the implementation area predicted by HYPER. It is widely recognized that the quality of the prediction model is inversely proportional to number of parameters used during the prediction model building [32]. The number of parameters is equal to the sum of the number of input variables in the model, and amount of data needed to describe the model. We built the interconnect capacitance model using only one variable as the predictor (estimated area of the chip) and the complexity of the curve which fits data was minimized as much as possible. Although it appeared that both the line and quadratic polynomial, and in particular the third order polynomial fit the data well, none of these models passed the strict statistical test for goodness of fit and resubstitution validation procedure. However, a piecewise linear least square fit showed both excellent accuracy and robustness.

The three segment piece-wise linear fit used to model the interconnect is characterized by 4 points, which are (2.95, 16.5), (10, 77.6), (23.5, 217) and (80, 1257). The model is valid from a predicted area of 2 mm^2 (which is equal to the size of a chip with one execution unit, with a few registers and interconnects) to a predicted area of 90 mm^2 (which is equivalent to a chip can that can accommodate more than 30 execution units, with more than hundred registers and multiplexers).

The measurement on all 50 examples, showed that the average error of the piecewise linear model is less than 18% percent, and the maximum error is smaller than 33%. Robustness of the model is illustrated by the fact that during 10 random resubstitution of 25 different values for prediction, the largest average error was below 20%, and the maximum error did not exceed 40%.

During the development of prediction tools, often consistency (for two different randomly selected instances, the one with lower predicted value indeed has lower measured value) is more important than accuracy. Consistency of our single prediction variable (predicted area) was, despite the very simple and robust model, higher than 96%. Figure 11 shows both the measured (for all 50 examples) and predicted values of the interconnect capacitance as a function of the estimated chip area.

Once the physical interconnect capacitance is accurately estimated, it is easy to establish a good interconnect power consumption model. The interconnect capacitance component is then given by:

$$C_{interconnect} = \alpha * C_{total} / N \quad \textbf{(EQ 8)}$$

where α is the average activity (the total number of interconnect accesses multiplied by an average signal transition probability), C_{total} is the total estimated interconnect capacitance of the chip and N is an estimate of the number of physical interconnects (after bus-merging). The HYPER system provides accurate estimates of the number of interconnects and activity.

The interconnect model was built using the automatic place and route features of the LagerIV placement and layout tool (Flint). In the future, a model should be developed based on hand-optimized floorplanning so the user can get feedback on the efficiency of their layout.

5.1.4 Control Logic

High level estimation of the power consumed by the controller, like that of interconnect, is complicated because the control is not defined until the hardware mapping process. Neither the number of control blocks nor the their function / size is known at the estimation stage, making it impossible to estimate any properties of the control theoretically. After scheduling, the control is defined and optimized by the *hardware mapper* and further by the *logic synthesis process* before mapping to layout. Like interconnect, therefore, the control needs to be estimated statistically.

The distributed control model used by HYPER is especially suited for low power. The control model incorporates a central finite state machine from which the state information is distributed to local controllers. Control signals (e.g. LOAD for the register file) for the datapath are generated in the local controllers. Bus capacitance on the control lines is reduced by placing the local controllers close to the datapaths. Thus, only the global state lines need to be distributed globally across the chip.

A large amount of data from several different DSP algorithms were analyzed in order to model the power consumed by the global and local controllers. Both initial and transformed versions were used to get a complete description of the sample space. Fully pipelined versions were not considered since there is only one control cycle per sample period, eliminating the need for a controller. Each of the benchmark examples were mapped to SDL (structural description) using HYPER and then into layout using the LAGER IV silicon compiler. IRSIM was used to extract the switching capacitances required to build the statistical model. This process has been automated for characterizing new libraries or the effects of new tools.

Global Control Model:

The amount of capacitance switched in the global controller was found by simulating each benchmark circuit for a whole sample period after the initial conditions were set up. The capacitance switched per sample period in the global controller was found to be directly related to the number of states, N_{states}, and is given for a 1.2 μm technology by:

$$C_{FSM} = \alpha_1 N_{states} + \alpha_2 \quad \text{(EQ 9)}$$

For a 1.2 μ technology, α_1 is 4.9fF and α_2 is 22.1fF. Figure 12 shows the total capacitance switched by the global controller as a function of the number of states. Note that the number of states not only determines the number of cycles the FSM goes through, but also the number of output bits it drives. Therefore, the total number of transitions and hence the capacitance switched, is strongly dependent on the number of states.

Local Control Model:

Since there are several local controllers in each design, the local controllers account for a larger percentage of the total capacitance than the global controller. A set of fifty examples were used to build the capacitance model for the local controller. Table 3 shows the data obtained for four different examples. The capacitance switched in the local controllers were determined by applying the state input sequence to each of the local controllers and simulating using IRSIM over a whole sample period. Correlations between the capacitance switched and several parameters like the number of states in the control, the number of outputs of the controller, and the total number of transitions on the output control signals were measured. The capacitance of the control was found to be highly correlated to the number of transitions on the output control signals. Note that, in this case, unlike the global controllers, the number of states gives no information about the number of transitions on the output nodes which depends on the glue-logic to be implemented. The transitions must therefore be separately accounted for. Using statistical tools to fit the data to a polynomial function, it was found that the total capacitance switched, in one sample period, for any local controller is a linear function of the number of transitions, the number of states and the bus factor given by (EQ 10).

$$C_{lc} = \beta_0 + \beta_1 N_{trans} + \beta_2 N_{states} + \beta_3 B_f \quad \text{(EQ 10)}$$

where N_{trans} is the number of transitions, N_{states} is the number of states, B_f is the bus factor (explained below), and C_{lc} is the capacitance switched in any local controller in one sample period. B_f represents the activity factor on busses and is defined as the ratio of the number of bus accesses to the number of busses. It is a measure of the average number of times busses are accessed. It represents a measure on the number of multiplexors and each multiplexor requires control signals from the controller. For a 1.2 μ technology, β_0, β_1, β_2 and β_3

are 72, 0.15, 8.3 and 0.55 respectively. Figure 13 shows the correlation between the actual and predicted capacitance switched per sample period for several different examples.

This model, though accurate to within 20% (as indicated by resubstitution validation procedure) is not sufficient since it assumes that the number of transitions is known. The number of transitions depends on assignment, scheduling, optimizations performed by the hardware mapper and the logic optimization tool(misII), the standard cell library used, the amount of glitching, and the statistics of the inputs. It is impossible to determine the combined effects of all these elements apriori. A statistical model relating the number of transitions to high level parameters was therefore derived. The important high level parameters that affected the number of transitions are:

- Size/complexity of the graph, i.e. the number of nodes and the number of edges. This is because the number of reads from and write to registers is dependant on the number of edges and nodes in the graph respectively. Control signals need to be generated for every read and write process.
- The number of execution units times the number of states. This is because each execution unit receives the clock and clock inverse every control cycle. In the hardware model used, the clock inverse is generated in the local controllers and fed to the execution units.

The HYPER high level synthesis system provides a lower bound on the number of execution units within as accuracy of 10% [28]. The lower bound on the busses tracks the actual number of busses closely and the maximum number of busses tracks the total number of bus accesses. The numbers predicted by HYPER were therefore used in building the correlation model. The fit function obtained for the total number of output transitions on local controllers based on the three high level parameters mentioned above is given in (EQ 11).

$$N_{trans} = \gamma_1 + \gamma_2\left(N_{nodes} + N_{edges}\right) + \gamma_3\left(S \times N_{Exu}\right) \quad \textbf{(EQ 11)}$$

where N_{trans} is the number of transitions on the outputs of the local controllers, S is the number of control cycles per sample period, N_{edges} and N_{nodes} are the number of edges and nodes respectively in the CDFG and N_{exu} is an estimate for the total number of execution units. For a 1.2 m technology γ_1, γ_2 and γ_3 are 178.7, 7.2 and 2.0 respectively. Figure 14 shows the total number of transitions in one sample period for the different examples vs. the fit function for the transitions based on high level parameters.

This statistical estimate takes into account the effects of logic synthesis and optimization. Effect of undetermined factors such as glitching are also included. Each datapath element in the cell library has a built in control-slice to locally buffer the incoming control signals. The internal gate and routing capacitance is taken into account when characterizing the leafcells. For example, the multiplexor select signal for a 16-bit datapath is buffered in the control slice. Therefore, the controller only drives a single (typically minimum sized) gate. This information was used to determine the loading capacitance during simulations.

5.2 Supply Voltage Estimation

The power supply voltage at which the flowgraph implementation will meet the timing constraints is estimated. The initial flowgraph which meets the timing constraints is typically assumed to be operating at a supply voltage of 5V with a critical path of $T_{initial}$ (the initial voltage will be lower if $T_{initial} < T_{sampling}$, where $T_{sampling}$ is the throughput constraint). After each move, the critical path is re-estimated, and the new supply voltage at which the transformed flowgraph still meets the time constraint, $T_{sampling}$, is determined. For example, if the initial solution requires 10 control steps (and let's assume that this is the same as the sampling period) running at a supply voltage of 5V, then a transformed solution that requires only 5 control steps can run at a supply voltage of 2.9V (where the delay increases by a factor of 2, Figure 1b) while meeting the same constraints as the initial graph.

A model for delay as a function of V_{dd} is derived from the curve shown in Figure 1b. Let the speedup of a transformed solution be defined as:

$$\text{Speedup} = T_{sampling} / T_{criticalpath} \quad \textbf{(EQ 12)}$$

where $T_{criticalpath}$ is the critical path of the transformed solution.

Since, the major power reduction during CDFG optimization using transformations is attributed to a reduction of the supply voltage and since the supply voltage has to be constantly evaluated (every time the objective function is called), it is important to model delay-V_{dd} relationship using an accurate and computationally efficient procedure. This relationship (of delay-V_{dd}) was modeled using Neville's algorithm for rational function interpolation and extrapolation [33]. Neville's algorithm provides an indirect way for constructing a polynomial of degree N-1 so that all of the used points are exactly matched.

Figure 15 also shows the accuracy of the interpolated data (plotted for speedup values ranging from 0.86 to 26 with increments of 0.1) when compared to the experimental data. Figure 16 shows an overview of the power estimation routine. It takes a control dataflow graph as input and computes the power using the existing estimation routines (critical path, bounds on resources and activity) along with the newly developed capacitance and voltage estimation routines (as discussed in this section).

6.0 Optimization Algorithm

The algorithm development for power minimization using transformations was greatly driven by several considerations and constraints. The basic strategy involved:

- Efficient tailoring of existing library transformations for power optimization.

- Fully utilizing the information about the design solution space obtained through extensive experimentation and theoretical insights (so that the algorithm can quickly come out of deep local minima).
- Efficient use of computer resources (CPU time and memory) so that large real-life examples can be addressed.
- Developing the modular software so that future enhancements or changes can easily be incorporated.

The computational complexity analysis of the power minimization problem showed that even highly simplified versions of the power optimization tasks using transformations are NP-complete. For example, we have shown that retiming for power minimization is a NP-complete problem. Computational complexity of retiming for power optimization is particularly interesting and somewhat surprising, since retiming for critical path reduction has several algorithms of polynomial-time complexity [34].

The computational complexity of the power minimization problem implies that it is very unlikely, even when the set of applied transformations is restricted, that polynomial time optimal algorithm can be designed. Two widely used alternatives for design of high quality suboptimal optimization algorithms are probabilistic and heuristic algorithms. Both heuristic and probabilistic algorithms have several distinctive advantages over each other. While the most important advantage of heuristic algorithms is a shorter run time, probabilistic algorithms are more robust and have stronger mechanisms for escaping local minima's.

An extremely wide spectrum of probabilistic algorithms are commonly used in various CAD and optimization problems. Among them, the most popular and successful are simulated annealing, simulated evolution, genetic algorithms, and various neural network algorithms (e.g. Boltzmann machine). Although recently there has been a considerable effort in analyzing these algorithms and the type of problems they are best suited for (for example a deep relationship between simulated annealing and solution space with fractal topology have been verified both experimentally and theoretically [35]), algorithm selection for the task at hand is still mainly an experimental and intuitive art.

In order to satisfy all major considerations for the power minimization problem, we decided to use a combination of heuristic and probabilistic algorithms so that we can leverage on the advantages of both approaches. Before we get into the algorithmic details, we will briefly outline the set of transformations used.

The transformation mechanism is based on two types of moves, global and local. While global moves optimize the whole DCFG simultaneously, local moves involve applying a transformation only on one or very few nodes in the DCFG. The most important advantage of global moves is, of course, a higher optimization effect; the advantages of local moves is their simplicity and small computational cost. We used the following global transformations (i) retiming and pipelining for critical path reduction (ii) associativity (iii) constant elimination and (iv) loop unrolling. In the library of local moves we have implemented three algebraic transformations:

associativity (generalized to include properties of inverse elements as introduced in [22]), commutativity, and local retiming. All global moves minimize the critical path using polynomial optimal algorithms. Their use is motivated by the need for shorter run-times. For example, the global pipelining move for a 7th order IIR filter with 33 nodes took 1.8 CPU seconds on a SPARC2 workstation. Extensive experimentation showed that although the best solution in semi-exhaustive searches is rarely one with the minimum critical path, it is very often topologically very close (a small number of moves is needed to reach it starting from a solution with the shortest critical path).

The Leiserson-Saxe algorithm for critical path minimization using retiming is used. The algorithm is modified in such a way that it automatically introduces the optimal number of pipeline stages needed to minimize the cost function for power. Pipelining is accomplished by allowing the simultaneous addition of the equal number of delay elements on all inputs or all outputs but not both. The Leiserson-Saxe algorithm is strictly based on minimizing the critical path and inherently does not optimize the capacitance being switched for a given level of pipelining. The modified algorithm determines the level of pipelining where the power is minimized and this is used as a “good” starting point for optimization using local moves (as will be discussed below). The global move for associativity involves the minimization of the critical path using the dynamic programming algorithm. Loop unrolling does not involve any optimization, instead it enables transformations which reveal large amount of concurrency for other transformations. For each local move, we defined the inverse local move which undoes the effect of the initial move.

As previously mentioned, the algorithm for power minimization using transformations has both heuristic and probabilistic components. While the heuristic part uses global transformations, the probabilistic component uses local moves. The heuristic part applies global transformations one at the time in order to provide good starting points for the application of the probabilistic algorithm. The probabilistic algorithm conducts a search in a vicinity of the solution provided by the heuristic part with the goal of minimizing the effective capacitance (trading-off control, interconnect and activity). The underlying search mechanism of the probabilistic part is simulated annealing with the goal that local minima’s can be escaped by using uphill moves and the effects of various transformations can be efficiently combined so as to minimize the capacitance component of power.

Initially constant propagation is applied to reduce both the number of operations and the critical path, followed by the modified Leiserson-Saxe algorithm and the dynamic programming algorithm for the minimization of the critical path using associativity. This step is followed by simulated annealing which tries to improve the initial solution by applying local moves probabilistically. For simulated annealing we used the most popular set of parameters [36]. For example, we used a geometric cooling schedule with a stopping criteria which

terminates the probabilistic search when no improvement was observed on three consecutive temperatures. After each move, a list of all possible moves is generated for each local transformation. The transformation and particular move to be applied is then selected randomly. In addition to the above algorithm, loop-unrolling can be used as preprocessing step.

HYPER-LP also allows optimization using a user specified sub-set of transformations (for example, optimizing with only pipelining). Figure 17 shows an overview of the HYPER-LP system.

7.0 Examples and Results

The techniques described in the previous sections will now be applied to “real-world” examples to demonstrate that a significant improvement in power can be achieved. Three examples of distinctly different computational structures are presented in this section and are optimized for power using the HYPER-LP system. Multiplications with constants were converted to shift-add operations. The improvement from this transformation was not taken into account since this it is a standard transformation performed for area or throughput optimization. The run time for all examples were less than 1 minute on a SPARC 2 workstation. The bitwidth requirement was established using the HYPER simulator tool [11].

7.1 Example #1: Wavelet Filter

The first example is a wavelet filter, used in speech and video applications. The implementation contains high-pass and low-pass filters that are realized as 14th order FIR modules using a wordlength of 16-bits (determined by high-level simulation). 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, retiming and pipelining provides an efficient and straight forward way to reduce critical paths (and thus voltage) while preserving throughput and the number of operations.

The initial reference design is a fully time-multiplexed area-optimized realization of the filter, with minimal amount of resources (1 adder, 1 subtractor, 1 shifter, and 4 global busses). The number of global busses were minimized using global bus-merging (i.e busses that are never accessed simultaneously can be collapsed) with the goal of minimizing the area. This solution has a critical path of 22 clock cycles and can be clocked at a maximum frequency of 1.5Mhz (since the critical path for this 16-bit datapath is around 30ns). Using HYPER-LP, it was found that Retiming by itself was sufficient to reduce the critical path of this design to 3 control cycles. Since the throughput requirement is 1.5Mhz, the clock period can be made approximately 7 times as

long (= 22 / 3) while meeting the timing constraint and therefore the supply voltage can be reduced to 1.5V (Figure 1b).

Since the amount of resources increases significantly (since the allocation is done with available time = critical path = 3 cycles), we expect the chip area and hence the interconnect component of power to increase. This is seen from Figure 18, which shows a plot of the average capacitance per bit obtained from layout extraction as a function of the bus number for the initial 22 cycle implementation (which contained 4 busses) and for the final 3 cycle implementation (which contained 25 busses). The figure also shows the capacitance obtained from the estimation of interconnect using the model presented in section 6.1.3.

The global bus capacitance switched increased by a factor of 3 from 400pF to 1200pF. However, the total capacitance switched was actually ***smaller*** than the minimal area version! There are several factors contributing to this decrease in capacitance:

- The amount of multiplexor, tri-state and control switches (which contributed to 45% of the total power of the minimal area design) were reduced significantly.
- The initial design uses bus-merging to significantly reduce the interconnect area. However, this does not come for free as the loading for the tri-state buffers driving the bus increases. Also all of the multiplexors whose inputs are connected to the shared bus will switch every time the bus value transitions, resulting in extra power. The problem is not as severe in the final design since there is little room for bus-sharing since there are fewer control cycles.
- The average transition activity (which is not modeled by the estimator presented in Section 6) and hence the capacitance per access for the logic modules (adder, subtractor, and shifter) was significantly lower in the final version. The reason is that in the initial version, since the modules were fully time-multiplexed, the inputs to the modules were coming from different sources (and hence uncorrelated) and had a higher probability of transitioning. In the final version (in which many units were allocated), there are longer periods of inactivity during which the inputs to the modules do not change and hence the units do not switch, resulting in lower activity.

The total power was reduced by a factor of 18 (these results are obtained from the IRSIM switch level simulator using a linear RC model using real input data). Table 4 shows the statistics of the initial and final designs. Note that area has been traded for lower power.

7.2 Example #2: IIR Filter

The second example is a 16-bit 7th order IIR filter with a 4th order equalizer designed using the filter design program Filsyn [37]. 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, when the transformation set of the HYPER-LP system was limited only to retiming, no reduction in the power consumption was observed. However, when the transformation set was enhanced to include both retiming and pipelining, the tool was very effective and achieved a reduction in the number of control steps used from 65 to 6 cycles. The implementation area increased as the number of control

cycles decreased. Figure 19a 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 for lower power. It is immediately evident that the price paid for reducing critical path is the increased registers and routing overhead. Figure 19b 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 increases at a very fast rate and the power starts to increase with further reduction in supply voltage. The power reduced approximately by a factor of 8 as a result of dropping the voltage from 5V to approximately 1.5V. Figure 20 shows a plot of Power vs. V_{dd} for a fixed throughput. Note that the low-power solution is somewhere between the minimal area solution and the maximum throughput solution.

7.3 Example #3: Volterra Filter

The third example is a second order Volterra filter. 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 power reduction only by a factor of 1.5). For this case, loop unrolling, retiming and associativity transformations proved to be important in alleviating the recursive bottleneck, allowing for a reduction in critical path and voltage.

Figure 21 shows the final layouts of optimizing the Volterra filter for power as well as for area while keeping the throughput fixed. This final implementation has approximately nine times better power characteristic and the same area as the initial implementation.

Table 5 shows a summary of power improvement after applying transformations relative to initial solutions that met the required throughput constraint at 5V for the representative examples described in this section. The results indicate that a large reduction in power consumption is possible (at the expense of area) compared to present-day methodologies. Also interesting was the fact that the optimal final supply voltage for all the examples was much lower than existing standards and was around 1.5V. A couple of overall comments on the results obtained:

- It is possible, after optimization, that the final solution will end up with a supply voltage that is non-standard (e.g 2 V). In this case, high efficiency (> 90%) switching regulators (as opposed to linear regulators) can be used to provide the required supply voltage and voltage conversion circuits can be used at the system level to communicate between subsystems that are working at different voltages. Voltage level conversion circuitry can be implemented with very low area and power overhead [38]. The use of multiple supply voltages (in which each part of the system operates at its own “optimum” voltage) can result in significant system power reduction compared to a solution which uses a single power supply voltage.

- Reducing the supply voltage raises some issues on noise immunity. Inductive noise is a result of the voltage drops (L * di/dt) across the inductors in the supply lines. Ignoring the threshold effects on current, we see that the di/dt scales as V^3 (even faster when threshold effects are taken into account), and hence drastically reducing the voltage noise. However, as mentioned earlier, reducing the voltage will require more parallelism (to meet throughput requirements) and will hence increase the number of simultaneously switching elements. Even so, the overall noise reduces in approximately a quadratic fashion with respect to the supply voltage. Also, using advanced packaging can reduce the lead inductances dramatically over conventional packaging (e.g a MCM package has less than 2nH of lead inductance while a typical wirebond pin grid array package has about 10nH). A similar analysis shows that the resistive noise (proportional to the current) scales at least linearly.

8.0 Conclusions

Power minimization is becoming a very important problem with the increasing demand for portable computing. An automated high-level synthesis system, HYPER-LP, was presented for optimizing power consumption in algorithm specific datapath intensive circuits using a variety of architectural and computational transformations. The synthesis approach consisted of applying transformation primitives in a well defined manner in conjunction with efficient high-level estimation of power consumption. The experimental results indicate that more than an order of magnitude reduction in power is possible over current-day design methodologies (that tend to minimize area for a given throughput) while maintaining the system throughput. It was found that the optimal supply voltage for minimizing power was much lower than existing standards (present-day 5V and emerging 3.3V) and was around 1.5V for most of the examples investigated. This work has addressed some key problems in the automated design of low-power systems, and provides a good starting point for addressing other research problems like detailed power estimation, module selection, partitioning, and scheduling for power optimization.

Acknowledgments

This project was funded by ARPA. We would like to thank several members of the HYPER team for their support and suggestions; including Shan-Hsi Huang for coding and supporting the library of local transformation primitives, Ole Bentz for supporting the back-end HYPER code (HardwareMapper), Ingrid Verbauwhede for many useful suggestions and the design and high-level simulation of the wavelet filter, and Scarlett Wu for the functional simulation of the wavelet filter at the IRSIM level.

References

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

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

[3] M. Kakumu and M Kinugawa, “Power-Supply Voltage Impact on Circuit Performance for Half and Lower Submicrometer CMOS LSI”, IEEE Transactions on Electron Devices, Vol 37, No. 8, pp. 1902-1908, August 1990.

[4] D. Dahle, “Designing High Performance Systems to Run from 3.3V or Lower Sources”, Silicon Valley Personal Computer Conference, pp. 685-691, 1991.

[5] K. Yano, et al., ‘‘A 3.8ns CMOS 16x16 Multiplier Using Complementary Pass Transistor Logic’’, IEEE Journal of Solid-State Circuits, pp. 388-395, April 1990.

[6] T. Callaway and E. Swartzlander, “ Optimizing Aritmitic Elements for Signal Processing”, VLSI Signal Processing Workshop, pp.91-100, 1992.

[7] A. Shen, A. Ghosh, S. Devedas, K. Keutzer, “On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks”, Proc. IEEE ICCAD, pp. 402-407, 1992.

[8] H. Trickey, “Flamel: A high-Level Hardware Compiler”, IEEE Transaction on CAD, Vol. 6, No. 2, pp. 259-269, 1987.

[9] R.A. Walker, D.E. Thomas, “Behavioral Transformation for Algorithmic Level IC Design” IEEE Trans. on CAD, Vol 8. No.10, pp. 1115-1127, 1989.

[10] B.S. Haroun, M.I. Elmasry, “Architectural Synthesis for DSP Silicon Compilers”, IEEE Transaction on CAD for IC, Vol. 8, No. 4, pp. 431-447, 1989.

[11] J. Rabaey, C. Chu, P. Hoang, M. Potkonjak, “Fast Prototyping of Data Path Intensive Architecture”, IEEE Design and Test, Vol. 8, No. 2, pp. 40-51, 1991.

[12] F.H.M. Franssen, F. Balasa, M.F.X.B. van Swaaij, F.V.M. Catthoor, H.J. De Man: “Modeling Multidimensional Data and Control Flow”, IEEE Trans. on VLSI Systems, Vol. 1, No. 3, pp. 319-327, 1993.

[13] M. Cirit, “Estimating Dynamic Power Consumption of CMOS Circuits”, IEEE International Conference on Computer Aided Design, pp. 534-537, 1987.

[14] F. Najm, “Transition Density, A Stochastic Measure of Activities in Digital Circuits”, DAC, pp. 644-649, 1991.

[15] A. Ghosh, S. Devedas, K. Keutzer, and J. White, “Estimation of Average Switching Activity”, Proc. DAC, 1992.

[16] N. Kimura, J. Tsujimoto, “Calculation of Total Dynamic Current of VLSI Using a Switch Level Timing Simulator (RSIM-FX)”, CICC, 1991.

[17] 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.

[18] P. E. Landman and J. M. Rabaey, “Power Estimation for High Level Synthesis,” Proceedings of the European Conference on Design Automation ‘93, Paris, France, Feb. 1993, pp. 361-366.

[19] K.K. Parhi: “Algorithm Transformation Techniques for Concurrent Processors”, Proc. of the IEEE, Vol. 77., No. 12, pp. 1879-1895.

[20] E. Feig, S. Winograd: “Fast Algorithms for Discrete Cosine Transform”, IEEE Trans. on Signal Processing, Vol. 40, No. 9, pp. 2174-2193, 1992.

[21] M. Potkonjak, J. Rabaey: “Maximally Fast and Arbitrarily Fast Implementation of Linear Computations”, IEEE ICCAD, pp. 304-308.

[22] M. Potkonjak and J. Rabaey, “Optimizing the Resource Utilization Using Transformations”, Proc. IEEE ICCAD Conference, pp. 88-91, November 1991.

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

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

[25] C. S. Shung et. al., “An Integrated CAD System for Algorithm-Specific IC Design.”, IEEE Transactions on CAD of Integrated Circuits and Systems, April 1991.

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

[27] M. B. Srivastava, M. Potkonjak: “Transforming Linear Systems for Joint Latency and Throughput Optimization”, EDAC-94, pp. 267-271, 1994.

[28] J. Rabaey and M. Potkonjak, “Estimating Implementation Bounds for Real Time Application Specific Circuits”, European Solid-State Circuits Conference, Milano, Italy, pp. 201-204, September 11-13, 1991.

[29] D. Schultz, “The Influence of Hardware Mapping on High-Level Synthesis”, U.C. Berkeley M.S. report, ERL M92/54.

[30] F.J. Kurdahi, A.C. Parker: “Techniques and Area Estimation of VLSI Layouts”, IEEE Trans. on CAD, Vol. 8, No. 1, pp. 81-92, Jan 1989.

[31] W.R. Heller et al. “Prediction of wiring space requirements for LSI”, IEEE/ACM Proc. 14th Design Automation Conference, pp. 20-22, 1977.

[32] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone: “Classification and Regression Trees”, Wadsworth & Brooks, Monterey, CA 1984.

[33] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling: “Numerical Recipes in C”, Cambridge University Press, 1988.

[34] C.E. Leiserson, J.B. Saxe, “Retiming Synchronous Circuitry”, Algorithmica, Vol. 6., No. 1, pp. 5-35, 1991.

[35] G.B. Sorkin, “Efficient Simulated Annealing on Fractal Energy Landscapes”, Algoritmica, Vol. 6, No. 3, pp. 367-418, 1991.

[36] Algoritmica, Special Issue, Simulated Annealing, A. Sangiovanni-Vincentelli, ed. Vol. 6, No. 3, pp. 295-478, 1991.

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

[38] A. P. Chandrakasan, A. Burstein, R. W. Brodersen, “A Low-power Chipset for Multimedia Applications”, IEEE International Solid-state Circuits Conference, pp. 82-83, February 1994.