

# An Integrated CAD Environment for Low-Power Design

**FOR MANY YEARS,** minimum area and maximum performance were the only two design criteria of any practical importance for commercial chips. Designers analyzed power consumption only as an afterthought, and used the results more to determine packaging requirements than to drive optimizations. Recently, however, power consumption has become increasingly important in determining the overall quality of a design. The principal driver for this has been the explosive increase in demand for portable electronics such as PDAs, laptops, and personal communicators. There has also been an increased push for low power in the high-performance computing market, motivated by reliability and cost issues associated with packaging and cooling high-power devices, such as DEC's 50W quad-issue Alpha microprocessor.

The surge of interest in low power has spawned numerous research efforts into design techniques for reducing power consumption.<sup>1</sup> The results of these studies have made it

PAUL LANDMAN

Texas Instruments

RENU MEHRA

JAN M. RABAEY

University of California,  
Berkeley

This CAD environment supports a high-level approach to power reduction, emphasizing optimizations at the algorithm and architecture levels of abstraction. An integrated set of analysis and optimization tools spans the design hierarchy, allowing the designer to make a systematic, top-down exploration and refinement of solutions in the area-time-power design space. In a case study—a low-power implementation of a digital bandpass filter—the CAD environment and tools yield more than an order of magnitude savings in power.

apparent that the most dramatic power reductions stem from optimizations at the higher levels of abstraction. In particular, designers have reported low-power strategies at the algorithm and architecture levels that promise orders of magnitude savings in power. (Published results based on gate and circuit level optimizations typically offer improvement by a factor of two or less.) This suggests we should take a top-down approach to low-power design. Specifically, optimization efforts should begin at the algorithm level, proceeding then to the architecture level, and finally to the gate, circuit, and layout levels of abstraction.

Considering the extremely low-power designs reported, it is evident that the difficulties facing the low-power community today do not stem from a lack of design techniques and methodologies. The methodologies are there; the problem is that there are no comprehensive CAD environments to support them. Existing CAD tools for low power offer point solutions to estimation and optimization.

For example, power analysis tools at the gate and circuit levels are widely available.<sup>2,4</sup> Logic synthesis tools targeted at low power are beginning to appear as well. Unfortunately, these tools fail to focus on the higher levels of abstraction where the most significant optimizations are possible.

There have been a few attempts at higher level analysis and optimization tools.<sup>5,6</sup> These tools tend to suffer from fairly severe inaccuracies, however, with error margins ranging from 50 to 100% or more. Moreover, these attempts fail to recognize that we cannot fully optimize a design by focusing on a single level of abstraction. Rather optimization efforts, and therefore tools, must span several levels of abstraction to produce the highest quality solutions.

The contribution of this article is a fully integrated CAD environment that supports a top-down methodology for low-power design. The framework targets digital signal processing (DSP) application-specific integrated circuits (ASICs), but could also serve as a model for more general environments. We demonstrate the efficacy of these tools and the CAD framework by applying them to a real-world design—in particular, the implementation of a low-power eighth-order bandpass filter. This example not only demonstrates the large power savings that result from high-level optimization techniques, but also illustrates how CAD tools can help in searching the vast area-time-power (ATP) implementation space.

### Design flow

We advocate a design flow that spans several levels of the design hierarchy, as shown in Figure 1. The suggested design flow begins at the algorithm level and proceeds down to layout. At each level, the designer is free to apply optimizations appropriate to that level. The following paragraphs take the reader through the design flow. We indicate the low-power techniques apropos to each

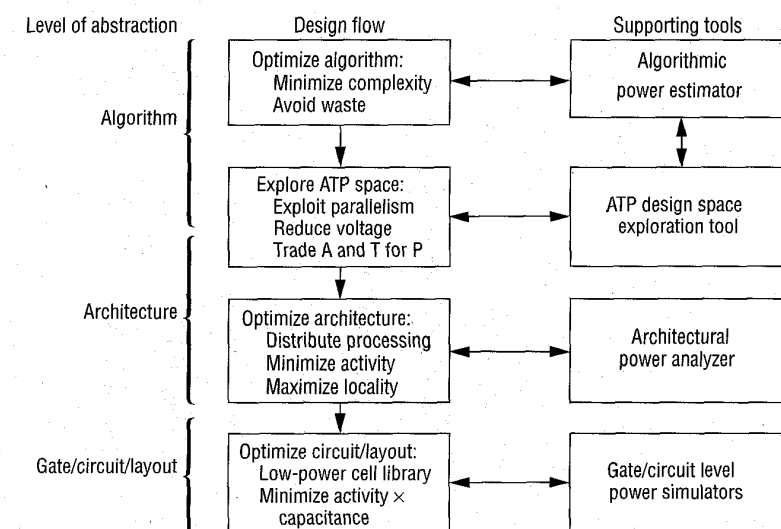


Figure 1. Design flow and supporting tools for low-power CAD framework. A, T, and P indicate area, time, and power.

step and describe how a hierarchy of analysis and optimization tools can converge on the desired low-power solution.

**Algorithm level.** In accordance with the top-down approach and the focus on DSP applications, the designer first explores algorithmic alternatives. Often, several available algorithms accomplish the same task but have quite different complexities. For example, Landman<sup>7</sup> describes three different speech-coding algorithms of approximately equal quality that have complexities differing by as much as 50%. Since it does not contribute to quality, this additional complexity is wasted. Avoiding waste is a recurring theme of low-power design.

The modularity or locality of the algorithm also has an important effect on power. Data transfers on global buses and accesses to global memories consume a large amount of power. Algorithms with a high degree of locality tend to map well onto more power-efficient distributed architectures with fewer global buses and memories. A useful CAD environment should contain a tool for evaluating the impact of

these algorithm level issues on power consumption. The CAD environment we propose contains an algorithm level power estimator to satisfy this need. Accurately predicting power and performance based solely on algorithmic criteria is a difficult problem, but by taking a library-based approach, the tool can produce estimates with a firm basis in reality.

Complexity and locality are not the sole predictors of power consumption. Many researchers have suggested that available concurrency is an equally important criterion, since it determines how well the algorithm maps to the popular low-voltage, parallel architectures.<sup>1</sup> Running processors at low voltage and exploiting parallelism to circumvent the loss in performance is sometimes referred to as trading area and performance for power. Our CAD environment contains a novel ATP exploration tool that allows the designer to rapidly make these trade-offs between the often conflicting requirements of complexity and concurrency.

Using the algorithm level estimation and exploration tools, the designer can

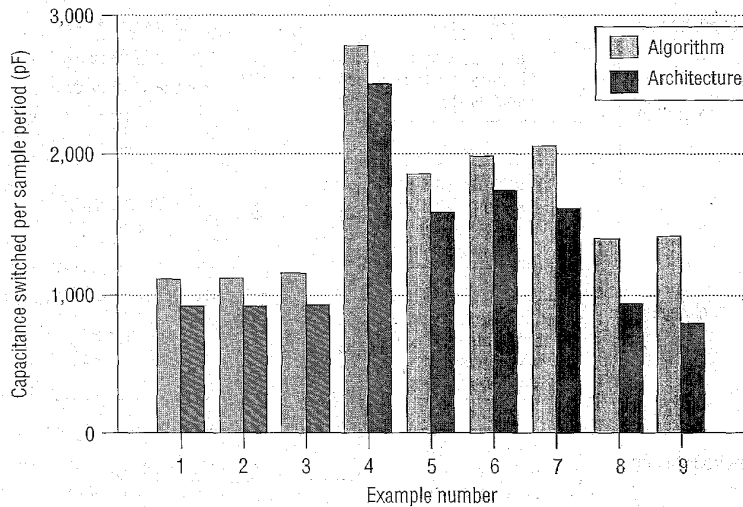


Figure 2. Correlation between the algorithm and architecture level power estimates.

begin to narrow down the ATP search space. The results from both tools, however, contain a good deal of uncertainty. The reason is that at this stage the chip structure and behavior are not fully specified, and this limits the accuracy possible at the algorithm level. Figure 2 exemplifies this point, showing the algorithmic and architectural power estimates for nine versions of a digital filter.

In general, there is adequate correlation between algorithmic and architectural estimates in spite of the lower absolute accuracy of the algorithm level tool. Therefore, estimates at the algorithm level can yield a relative evaluation of different designs. One must bear in mind, however, the limited accuracy of the algorithm level and make decisions based only on significant differences in algorithmic estimates. For example, in this case a designer relying solely on the algorithmic estimates might assume that versions 1, 2, and 3 were lower power than versions 8 and 9. The architectural estimates show that just the opposite is true. When algorithmic estimates are closely spaced, the designer should use architectural analysis to produce a finer grain, more reliable classification.

Therefore, an appropriate strategy is to use the algorithm level estimation and ATP exploration tools to get a rough ranking of algorithms and architectures. The designer can then narrow the search space down to a few candidates that seem to offer good performance (within the confidence interval of the estimation tools) and analyze them more carefully at the architecture level. By integrating tools at several levels of abstraction, our design environment makes such an approach possible.

**Architecture level.** After exploration at the algorithm level, the designer can consider optimizations at the architecture level. Since architectural analysis tools have more information available to them, they can provide the accuracy the designer requires to select the best algorithmic and architectural solution. Moreover, tools at this level can analyze phenomena that are transparent to the higher level estimation tools. For example, the assignment of operations to hardware units affects the signal statistics and activity of the architecture and, therefore, its power consumption. The assignment of operations from a temporally and spatially

local portion of the computational graph onto the same processing elements can yield maximum correlation and, therefore, minimum activity and power. We refer to this low-power technique as exploiting locality. While the algorithmic model is oblivious to such effects, architectural power analysis allows these kinds of refinements.

#### Gate, circuit, and layout levels.

After making algorithmic and architectural selections, the designer proceeds to gate, circuit, and layout level implementation, where further power optimizations are possible. For instance, the designer can supply the synthesis tools with a low-power cell library.<sup>8</sup> Also, chip placement and routing can target minimization of the activity-capacitance product of wires.<sup>9</sup> Moreover, as a final verification step, switch or device level analysis tools (such as PowerMill<sup>2</sup> and SPICE<sup>3</sup>) can validate the results of the higher level tools. To support the design flow down to this low level of abstraction, we have linked our CAD framework seamlessly to the Lager IV silicon compilation system.<sup>10</sup>

**The top-down approach.** To review, the methodology we propose incorporates a top-down approach to design optimization. Beginning at the algorithm level, the designer can invoke behavioral power estimators to begin to classify alternative algorithms in terms of intrinsic power requirements. ATP exploration tools can then evaluate the suitability of algorithms for implementation on low-power (for example low-voltage, concurrent) architectures. Next, the designer can use architectural power analysis to verify design decisions made at the algorithm level and explore additional power optimization opportunities that are transparent to algorithmic estimation tools. At all phases, power predictions use data from precharacterized cell libraries, allowing the user some level of

confidence in even the highest level estimates. Finally, the designer can synthesize the optimized algorithm and architecture down to layout and verify them with low-level analysis tools.

### Low-power CAD environment and tools

We built our low-power CAD framework around Hyper, a high-level synthesis tool targeted at generating ASICs for data-path-intensive DSP applications. Previous publications have described various aspects of the Hyper system.<sup>5,11-13</sup> Therefore, we will focus on recent extensions that allow the user to perform the kind of multilevel optimization and exploration we described earlier.

As shown in Figure 3, Hyper relies on three main tools to facilitate low-power design: an algorithmic power estimator, an ATP exploration tool, and an architectural power analyzer. All these tools employ a library of power models for data paths, memory, control, and interconnects. Each tool uses whatever information is available at that level of abstraction. By applying these tools in an integrated, top-down fashion, the user can begin with a high-level description of the desired functionality and systematically converge to the optimum low-power algorithm and architecture.

#### Algorithmic power estimation.

This process predicts the power consumption of a chip's data path, memory, control, and interconnect components, given only the algorithm it will execute. This is not an easy task, since the same operations can consume different amounts of power when performed on different pieces of hardware.

We can roughly estimate the data path and memory power consumption by looking at the type, quantity, and characteristic energy of the various operations the algorithm requires. To some extent, these characteristic energies depend upon the final hardware implementation; however, we can base

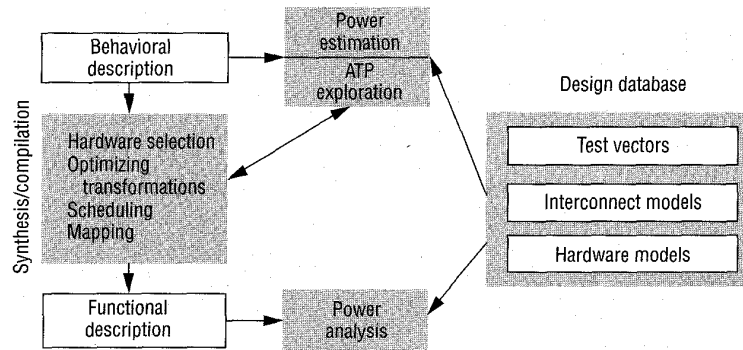


Figure 3. Hyper low-power design environment. Shading indicates extensions to facilitate low-power design.

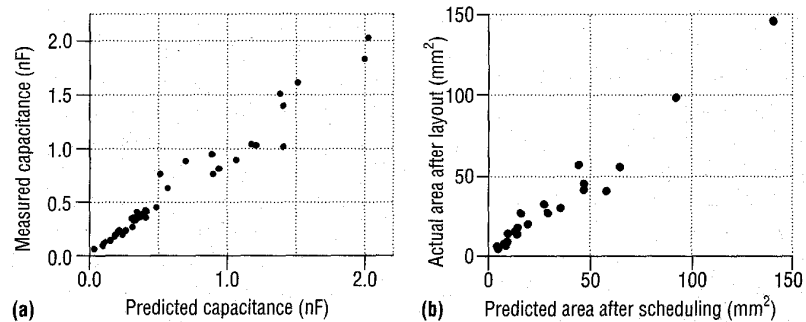


Figure 4. High-level prediction versus chip measurement: controller (a) and interconnect (b) models.

rough estimates on results from previous designs, perhaps, or from existing hardware libraries. For instance, Hyper selects memory and execution units from a custom, low-power hardware library. Since the library is predefined, the environment can characterize the power consumption of each cell a priori, reducing the data path power estimation task to a series of table lookups.

Thus, operation and memory access counts can help us estimate data path and memory power consumption based solely on the algorithm description. However, it is difficult to estimate controller and interconnect power without more detailed, implementation-specific information such as hardware allocations and chip area. By analyzing the topology of the behav-

ioral flow graph, including data dependencies and timing constraints, it is possible to produce reasonable estimates of these factors.<sup>11</sup> Combining these estimates with statistical models based on past designs, algorithmic estimators can provide meaningful early predictions of control and interconnect power consumption.

These models might take the form of a database relating the controller power or the average interconnect length of previous chips to algorithm level parameters. Hyper's algorithmic power estimator employs this approach. We extracted a statistical controller power model from a set of 46 benchmarks representing a wide cross section of DSP applications.<sup>14</sup> Figure 4a depicts the accuracy of the resulting model. The av-

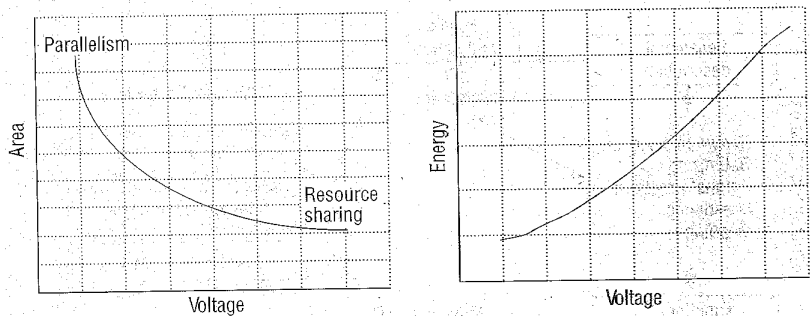


Figure 5. Qualitative area-power exploration graphs.

erage and maximum modeling errors for the benchmark set are 11.5 and 41.1%. We used a similar strategy to develop a high-level area model (see Figure 4b), which can be used to estimate average interconnect length. The average and maximum errors of this model are 17 and 44%.

The principal advantage of algorithm level estimation is that it provides the user with valuable feedback very early in the design process. Not surprisingly, this comes at the cost of accuracy, since the tools have access to only a very small amount of implementation information. For example, without knowledge of the final architecture and how hardware resources are shared, we cannot determine the activity statistics of signals and circuits. Consequently, the modules are characterized for completely random uniform-white-noise inputs. In many cases, the white-noise models are sufficient, with errors as low as 10 to 20% relative to switch level simulations. However, when data streams are correlated, estimation errors grow, and results can be off by 50 to 100% or more (see Figure 2).<sup>15,16</sup> We can eliminate these inaccuracies at the architecture level of analysis.

#### ATP design space exploration.

While estimation allows the designer to plot a single point in the ATP space, it is also important to explore a whole range of trade-offs between area, time,

and power. As mentioned earlier, one way of trading area and performance for power is to scale down voltage and make up for the loss in performance by employing techniques such as parallel processing. The Hyper environment provides an exploration tool that allows the designer to see this trade-off between area and power graphically.

The tool's output is a set of exploration curves that plot the estimated area and power of an algorithm as a function of supply voltage (see Figure 5). Typically the area increases for lower voltages, since the design must compensate for longer circuit delays with additional parallel processors (with their associated interconnects, memory, and control). This parallel processing allows the design to meet the algorithmic performance requirements while operating at reduced voltage.

The exploration produces the ATP curves point by point, by iteratively invoking the algorithmic power estimator over a range of operating voltages.<sup>14</sup> The curves provide useful feedback that can guide the designer in selecting the algorithm and architecture that offer the best compromise between area, performance, and power. As the exploration curves are based directly on the algorithmic power estimates, they are subject to the same inaccuracies. For a finer grain classification of implementations, we must rely on an architectural power analysis tool.

#### Architectural power analysis.

Prior to specifying an architecture, very little implementation-specific information is available to the estimation tools. This limits the accuracy that these tools can achieve. Architectural power analysis can obtain much better results, since the allocation, partitioning, and interconnection of data path, memory, and control modules are more completely specified.

For example, it was difficult to model power consumption at the algorithm level due to lack of knowledge about the activity statistics of the inputs feeding the various modules. In contrast, at the architecture level we can ascertain the activity by performing fast and efficient functional simulation of the design, perhaps using register-transfer level VHDL as a simulation platform. Then, we can generate accurate power estimates by applying black-box models that calculate module power consumption as a function of these activity measurements. So, instead of characterizing modules only for uniform white-noise inputs, we could characterize modules for many different types of input activity.

The Hyper architectural power analyzer, for example, uses this strategy to accurately estimate data path and memory power consumption. We based the model on the realization that two's-complement data consists of two types of bits, which exhibit quite different activity patterns: The least significant bits (LSBs) contain data, and the most significant bits (MSBs) contain sign. Figure 6 depicts the distinct activity behavior of the two bit types. The figure displays 0-to-1 bit transition probabilities for data streams with varying temporal correlations, labeled  $p$ . The uniform-white-noise activity model works well only for LSBs. The MSBs, or sign bits, have a much different activity, which depends intimately on the temporal correlation of the data stream. Positively correlated data streams result

in reduced sign activity, while negatively correlated streams have higher activity.

This dual-bit-type (DBT) model characterizes library cells not only for white-noise inputs, but also for various combinations of sign inputs. For example, the table of capacitive coefficients required to characterize a one-input module would contain the following entries:

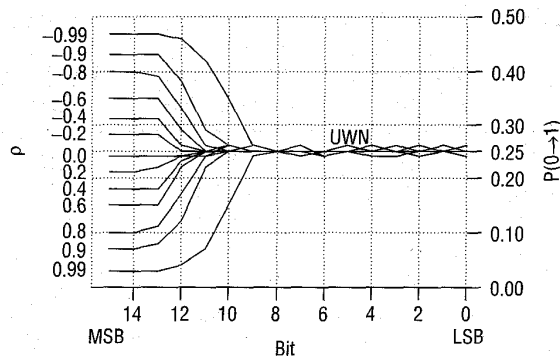
- uniform
 

white noise	$C_{UU}$
■ sign	$C_{++}, C_{+-}, C_{-+}, C_{--}$

$C_{UU}$  describes the capacitance switched during random (uniform white-noise) input transitions. In contrast,  $C_{+-}$  denotes the capacitance switched when the input changes sign from positive to negative. The other entries have similar interpretations. If desired, we can automate the process of generating the required input patterns and characterizing the library modules. DBT power estimates are typically within 10 to 15% of switch level simulations for all inputs, while the accuracy of the white-noise model depends on how well the inputs match white-noise assumptions.<sup>15,16</sup>

The difficulty with architectural power analysis stems primarily from lingering uncertainties as to the final placement and routing of the register-transfer level components. This lack of information makes it difficult to accurately estimate interconnect power consumption. Possible solutions to the dilemma include using interconnect models based on derivatives of Rent's Rule or back annotation after early floorplanning.

Overall, Hyper's architecture-level power estimates are typically within 20% of switch level simulations based on extracted layouts. Since the architectural power analysis tool provides such accurate results, we use it in this article as a basis of comparison for high-



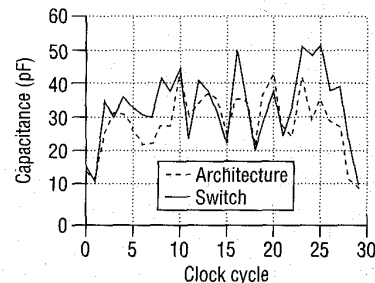
**Figure 6.** Transition activity versus bit for typical data streams: 0-to-1 bit transition probabilities for data streams with varying temporal correlations, labeled  $\rho$ . Positively correlated data streams result in reduced sign activity, while negatively correlated streams have higher activity.

er level estimates. Figure 7 provides some justification for this level of confidence, demonstrating that the architectural power analyzer provides results within 18% of switch level simulations for a sub-band speech-coding chip.

### Case study: The Avenhaus filter

How do we use these high-level analysis and exploration tools to support a comprehensive low-power design methodology? The general approach we've taken so far applies to a wide variety of applications. One example is the task of producing a low-power implementation of the Avenhaus filter.<sup>17</sup>

In this case study, we proceed through a sample design flow, at each stage highlighting issues of particular importance. We begin the process with a preliminary evaluation of the filter, comparing various structures we can use to implement its transfer function. Next, we explore the design space at the algorithm level and apply several of the low-power strategies mentioned earlier. After narrowing down the design space, we proceed to architecture level analysis to verify and refine our design decisions. We conclude with a review of the power savings achieved at each stage of optimization.



**Figure 7.** Architectural power analysis versus switch level simulation.

**Preliminary evaluation.** The Avenhaus, an eighth-order bandpass filter, can have several different structural implementations. We consider the cascade, continued fraction, direct form II, ladder, and parallel forms proposed by Crochiere and Oppenheim.<sup>17</sup> Each of these forms has a very different computational structure and, thus, should map to a distinct point in the ATP design space. Other studies have considered the algorithm level area and performance trade-offs for these structures,<sup>12</sup> but to the best of our knowledge ours is the first attempt to study the power aspects.

We'll assume the designer is free to

**Table 1.** Complexity of the initial structures operated at 5V supply voltage.

Filter type	Critical path (ns)	Max. sample frequency (MHz)	Word length (bits)	No. of multiplications	No. of adds	Energy (nJ)
Cascade	340	2.94	13	13	16	27.7
Continued fraction	950	1.05	23	18	16	89.5
Direct form II	440	2.27	19	16	16	59.5
Ladder	1,224	0.82	14	17	32	52.1
Parallel	306	3.27	13	18	16	36.1

**Table 2.** Complexity of the structures after constant multiplication expansion at 5V.

Filter type	Critical path (ns)	Max. sample frequency (MHz)	No. of adds	No. of subtractions	No. of shifts	Energy (nJ)
Cascade	361	2.77	38	23	51	43.5
Continued fraction	1,104	0.91	68	50	116	98.7
Direct form II	440	2.27	54	40	91	95.6
Ladder	1,406	0.71	36	31	46	75.4
Parallel	437	2.29	40	30	63	61.3

select the algorithm (the filter structure), apply any number of transformations to it, and choose an appropriate supply voltage. However, the design must meet a 2.75-MHz minimum throughput requirement imposed by the surrounding system.

As a preliminary step in the algorithm selection process, we profile each filter structure in terms of several key parameters that will influence power consumption. Table 1 shows the maximum throughput of each algorithm (in terms of critical path and sample frequency) and the complexity (in terms of required word length and operation count). It also gives an estimate of the energy required per sample, assuming a straightforward implementation of each structure. To allow a direct comparison, we quote maximum frequency and energy results for standard 5V implementations, with clocking frequencies set so that all operations finish within a single clock cycle. The behavioral estimator described earlier produced all estimates.

As we discussed earlier, the complexity of an algorithm has a major impact on the power consumed. Extra bits of word length contribute to larger physical capacitance and increased activity. Likewise, higher operation counts contribute directly to increased activity and can also necessitate increased hardware and routing, resulting in larger physical capacitance. Table 1 reflects this, showing that the cascade and parallel implementations have among the lowest operation counts and smallest word lengths and, consequently, the lowest energies. This relates directly to the previously mentioned low-power theme of avoiding waste.

Complexity is not, however, the only factor determining power consumption. The minimum operating voltage also has an important effect. Notice that, in their current forms, only the cascade and parallel structures can meet the 2.75-MHz throughput constraint at 5V. The critical paths for the other algorithms are much longer and, thus, cannot realize the required sample rate, even at 5V.

As yet, however, we have only considered the direct implementation of each filter type. Quite possibly, transformations such as constant multiplication expansion or pipelining could reduce one of the other algorithm's critical path, allowing it to operate at a much lower voltage and, perhaps, at lower power. Likewise, other transformations might drastically reduce the complexity of a structure, making it optimal for power. At this stage the cascade and parallel forms look promising, but we need more exploration before making a final selection.

**Programmable versus dedicated hardware.** One transformation that can be very useful in power reduction is expansion of constant multiplications into adds and shifts. The multiplication operation can then be implemented on one or more adders and programmable shifters. In this way, the hardware performs only additions and shifts corresponding to 1s in the coefficient. In contrast, the array multiplier imple-

mentation performs an addition for each bit of the coefficient, even if it is a 0. Therefore, if the coefficients have a relatively large number of 0s, it may be reasonable to use add-shifts since the array multiplier would perform unnecessary add operations.

On the other hand, the dedicated array multiplier has certain advantages. First, it performs shifts by hard-wired routing, while the add-shift version uses programmable shifters, latching intermediate results between stages. These shifters and latches contribute additional overhead and can offset the gains from the reduced number of additions. Whether this overhead is dominant or not depends on the particular value of the coefficient. (Actually, we can even consider optimizing or scaling coefficients for minimum power.)

In Avenhaus filters, we see an increase in energy after applying this transformation (see Table 2). While there are no multiplications, the number of additions has increased substantially, and several subtract and programmable-shift operations are now necessary. This increased complexity has a significant impact on the power consumed. Moreover, the programmable hardware has additional control overhead.

Table 2 also reveals considerable increases in some of the critical paths. This is a considerable penalty since, as we have noted earlier, a high-speed design is always desirable because of its potential for voltage reduction. For this example, therefore, we choose to use dedicated multipliers for each multiplication instead of programmable add-shifts. Again, this may not always be the optimum decision, and the designer must evaluate the trade-offs on a case-by-case basis.

**Critical-path reduction and voltage scaling.** Supply voltage reduction is an excellent way to reduce power consumption. As is apparent from Table 1, however, not all the algorithms

**Table 3.** Critical paths (in ns) after pipelining to different extents.

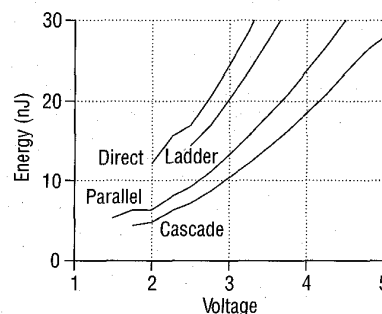
Filter type	Original critical path (ns)	Critical path (ns) with pipelining stages				
		1	2	3	4	5
Cascade	340	170	136	102	—	—
Continued fraction	950	850	—	—	—	—
Direct form II	440	132	—	—	—	—
Ladder	1,224	612	432	324	252	216
Parallel	306	170	102	—	—	—

can meet the throughput constraint even at 5V. Therefore, in their current forms we cannot consider reducing the voltage for any of these structures below 5V. If we can reduce the critical paths, however, we open the possibility of voltage reductions.

Graph transformations provide a powerful tool for design optimization at the algorithm level and are useful for critical-path reduction. Transformations alter the graph structures while preserving the input-output relationships. Our high-level design environment automates the task of applying transformations. This allows us to explore the effect of many different transformations on each of the candidate algorithms—a task that would not be practical for a designer taking a manual approach.

One important transformation for critical-path reduction is pipelining. By allowing more operations to occur in parallel, pipelining reduces the critical path of the design, enabling voltage reduction while still maintaining the required throughput. As a result of pipelining, some of the filter structures that could not meet the throughput constraint initially become feasible, while those already feasible can be made to operate at lower voltages than before.<sup>5</sup> Table 3 shows the reduction in critical paths after pipelining.

But critical-path reduction is only an indirect measure of improvement. We are actually interested in the minimum voltage and energy achieved after



**Figure 8.** Voltage reduction (and its effect on energy) after optimal pipelining.

pipelining (as shown in Figure 8). We produced the curves in Figure 8 using our exploration tool. These curves graphically illustrate that pipelining can appreciably reduce voltages (and energies).

Table 4 (next page) summarizes the results from the exploration curves. Notice that the optimum level of pipelining is not always equal to the maximum pipelining. This is due to the overhead associated with pipelining. For example, plotting the exploration curves for the maximally pipelined cascade would reveal that it consumes a minimum energy of 5.1 nJ, which is higher than the two-stage version, which consumes 4.4 nJ.

Clearly, the cascade and parallel versions still yield the best solutions. Therefore, based on the results obtained from our high-level exploration tools, we can at this point eliminate the

**Table 4.** Effect of optimal pipelining on the energy and area of the designs.

Filter type	No. of stages	Critical path (ns)	Minimum voltage	Energy (nJ)	Area (mm <sup>2</sup> )
Cascade	2	136	1.75	4.4	157
Continued fraction	1	850	—	—	—
Direct form II	1	132	2.00	12.3	374
Ladder	5	216	2.50	14.5	126
Parallel	2	102	1.50	5.3	411

continued-fraction, direct-form, and ladder implementations from consideration. We will not eliminate the parallel form at this stage since it gives results close to the cascade, and the error inherent in the high-level estimation tools does not allow us to resolve differences that are so small.

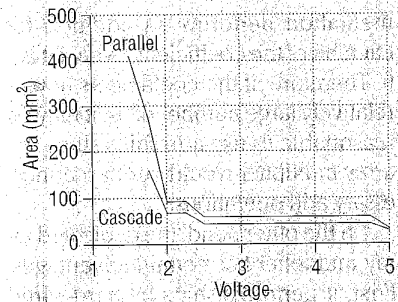
Power is not, however, the only consideration. As we reduce voltage we require more parallel hardware to meet the throughput constraints. So, while energy can be reduced by this technique, we must pay a price in terms of area. The area exploration curves of Figure 9 illustrate this point. Since the cost of a component is directly related to the amount of silicon real estate it requires, real-world designers have a limited amount of silicon area available. For our example, to achieve the minimum energies, the designer would have to accept significant area penalties as shown in the Figure 9. But with a slightly higher operating voltage (about 2V), the area penalties are much less severe and, moreover, the resulting energies are not significantly higher (see Figure 8). The exploration curves allow the designer to evaluate the area penalties associated with parallelism and make appropriate decisions on the area-energy trade-offs. For this example, let us assume that we choose to avoid the large area penalties by operating at 2V.

There are, of course, other power reduction techniques that the designer could explore within the Hyper environment. Chandrakasan et al.<sup>5</sup> describe

several transformations for low power. But the purpose of this case study is to present a general methodology and show how high-level tools can be used to facilitate low-power design—not to enumerate all possible implementations of the Avenhaus filter.

So far, we have limited the optimization process to algorithmic exploration and analysis. There are additional techniques at the architectural level for reducing power. Also, architectural power analysis will allow us to verify and refine the decisions made at the algorithm level.

**Architectural exploration.** Using power estimation and exploration tools, we've narrowed down the design space to two filter structures: cascade and parallel, each with two stages of pipelining operating at 2.75 MHz and 2V. Table 5 gives a detailed breakdown of the energy estimates at the algorithm and architecture levels. As described earlier, at the algorithm level we are unable to account for the effect of signal statistics. If we assume that we are filtering speech data (which is highly correlated) we might expect some inaccuracy in the algorithmic power estimates. To verify and refine these results, we now use Hyper to synthesize the selected algorithms and resort to architectural power analysis for more accurate energy estimates. Table 5 contains the results of this process as well. (We will explain the meaning of the local assignment column shortly.)

**Figure 9.** Area-energy trade-off for the cascade and parallel versions.

For a speech input, we see that the algorithmic power estimator did indeed overestimate the power consumed by the execution units by 47% (cascade) and 60% (parallel). However, the total chip powers were more accurate, with 17 and 23% errors. The important point, however, is that the errors in the algorithmic power estimates were systematic overestimates rather than random errors. This suggests that relative classifications made during algorithmic design space exploration are meaningful.

Also, the cascade structure continues to be the lowest power solution. Therefore, based on these accurate architecture level estimates, we can select the cascade filter as our final low-power Avenhaus implementation. Some final optimizations are still possible, however, at the architecture level.

For instance, assignment of operations to hardware operators can have a significant impact on power. One possible assignment strategy involves assigning operations to operators to maximize locality, as mentioned earlier. This is beneficial because signals that are local to a small subsection of the filter tend to be more highly correlated than widely separated signals. If we assign the operations on highly correlated signals to the same hardware units, there is likely to be less activity in the sign bits. This, in turn, should reduce the power consumption of the implementation. Note that the analysis of this effect can

only be performed using the DBT-based architectural power analyzer—an estimator based on the uniform white-noise model would not be able to make this distinction. The results of assignment for locality on the cascade also appear in Table 5. There is an overall additional energy reduction of 22%, with no penalty in area or performance.

**Gains from design space exploration.** This case study demonstrates that design decisions at the algorithm and architecture levels have a tremendous impact on a design's power. Selecting the correct algorithm (cascade, in this case) can save a factor of three in power, compared to the worst case (direct form). Moreover, the direct form (at 89.5 nJ) could not achieve the required 2.75-MHz sampling rate. If the algorithms were compared for the same throughputs, the cascade would actually be even more than three times better. We also found that, counter to what we may expect, expanding multiplications into shifts and adds is not universally beneficial and actually increases the power consumed for some cases. We found that transformations on the graph structure—for example, pipelining—further reduced the power by a factor of more than six times. Finally, local assignment helped reduce the power by another 22%. As Table 6 illustrates, coupling the algorithmic improvements with architectural optimizations allows us to achieve more than an order of magnitude power reduction (27 times for this example).

**OUR TOOLS,** implemented as part of the Hyper high-level synthesis system, facilitate a hierarchical, top-down approach and form an interactive framework for low-power design.

This work is ongoing, and other issues must still be addressed. Aside from improvements in the current power-modeling strategies, we need further re-

**Table 5.** Breakdown of the total energy for the cascade and parallel versions (with two pipeline stages).


Hardware class	Energy for cascade form (nJ)			Energy for parallel form (nJ)	
	Algorithm	Architecture	Local assignment	Algorithm	Architecture
Exu	2.42	1.65	1.44	3.27	2.05
Registers	0.59	0.50	0.50	0.64	0.52
Control	0.62	0.62	0.62	0.73	0.73
Buffers	0.08	0.06	0.07	0.09	0.06
Mux	—	0.18	0.18	—	0.21
Bus	1.01	1.04	0.29	1.31	1.31
Clock	0.31	0.25	0.25	0.38	0.33
Total	5.03	4.30	3.35	6.42	5.21

**Table 6.** Energy reductions from design space exploration. UWN indicates uniform white noise.

Power reduction technique	Inputs	Volts	Energy (nJ)	Energy reduction factor
Worst algorithm (direct form)	UWN	5.0	>89.5	—
Best algorithm (cascade)	UWN	5.0	27.7	3
After constant multiplication expansion on cascade	UWN	5.0	43.5	2
Pipelining (no area constraint)	UWN	1.5	4.4	20
Pipelining (with area 100 mm <sup>2</sup> )	UWN	2.0	5.0	18
Architecture level estimate	Speech	2.0	4.3	21
Assignment for locality	Speech	2.0	3.3	27

search into strategies for ordering transformations to enable maximum power reductions. In addition, the current environment is interactive and primarily user driven. In the future, one can envision a fully automated environment that intelligently searches the design space, attempting to optimize some cost function based on area, performance, and power.

In conclusion, the high-level design

space is vast and presents the designer with numerous trade-offs and implementation options. The multiple degrees of freedom preclude a manual exploration of all avenues for power optimization. The Hyper design space exploration environment provides the user with a means for rapidly and efficiently searching the design space for solutions that best meet area, performance, and power constraints. 

## Acknowledgments

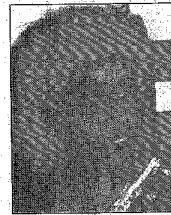
A fellowship from the National Science Foundation and ARPA grant J-FBI 93-153 sponsored this research. We gratefully acknowledge the support of these agencies.

## References

1. A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-Power CMOS Digital Design," *IEEE J. Solid-State Circuits*, Apr. 1992, pp. 473-484.
2. A. Deng, "Power Analysis for CMOS/BiCMOS Circuits," *Proc. Int'l Workshop Low Power Design*, 1994, pp. 3-8.
3. L.W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," Tech. Report ERL-M520, University of California, Berkeley, Electrical Engineering and Computer Science Dept., 1975.
4. A. Salz and M. Horowitz, "IRSIM: An Incremental MOS Switch-Level Simulator," *Proc. 26th Design Automation Conf.*, Computer Society Press, Los Alamitos, Calif., 1989, pp. 173-178.
5. A. Chandrakasan et al., "Optimizing Power Using Transformations," *Trans. on CAD*, Vol. 14, No. 1, Jan. 1995, pp. 1-12.
6. S.R. Powell and P.M. Chau, "Estimating Power Dissipation of VLSI Signal Processing Chips: The PFA Technique," *VLSI Signal Processing IV*, 1990, pp. 250-259.
7. P. Landman, *High Quality, Low Bit Rate Speech Coding for Low Power VLSI Implementation*, master's thesis, University of California, Berkeley, 1991.
8. T. Burd, *Low-Power CMOS Library Design Methodology*, master's thesis, University of California, Berkeley, 1994.
9. K. Chao and D. Wong, "Low Power Considerations in Floorplan Design," *1994 Int'l Workshop on Low Power Design*, 1994, pp. 45-50.
10. *Anatomy of a Silicon Compiler*, Kluwer, R. Brodersen, ed., Boston, 1993.
11. J.M. Rabaey et al., "Fast Prototyping of Datapath-Intensive Architectures," *IEEE Design & Test of Computers*, Vol. 8, No. 2, Jun. 1991, pp. 40-51.
12. M. Potkonjak and J. Rabaey, "Exploring the DSP Algorithm Design Space Using Hyper," *VLSI Signal Processing*, 1993.
13. J. Rabaey and L. Guerra, "Exploring the Architecture and Algorithmic Space for Signal Processing Applications," *Technical Digest of Int'l Conf. VLSI and CAD*, 1993, pp. 315-319.
14. R. Mehra and J. Rabaey, "Behavioral Level Power Estimation and Exploration," *Proc. Int'l Workshop on Low Power Design*, 1994, pp. 197-202.
15. P. Landman, *Low-Power Architectural Design Methodologies*, doctoral dissertation, University of California, Berkeley, 1994.
16. P. Landman and J. Rabaey, "Architectural Power Analysis: The Dual Bit Type Method," *IEEE Trans. VLSI Systems*, Jun. 1995, pp. 173-187.
17. R. Crochiere and A. Oppenheim, "Analysis of Linear Networks," *Proc. IEEE*, Vol. 63, No. 4, 1975, pp. 581-595.



**Paul Landman's** research at Berkeley focused on low-power digital design techniques and tools with an emphasis on DSP applications. After completing his dissertation, Landman joined the low-power design branch of the DSP R&D Center at Texas Instruments in Dallas. His initial research there focused on low-power algorithms, architectures, and circuits for portable video applications. Currently, he is participating in the development of a low-power programmable DSP targeted at wireless communication applications. Landman received BS, MS, and PhD degrees in electrical engineering and computer science from the University of California, Berkeley. He is a National Science Foundation Fellowship recipient and a member of Tau Beta Pi, Eta Kappa Nu, and the IEEE.



**Renu Mehra**, a PhD candidate at the University of California, Berkeley, is currently working on high-level power estimation and synthesis techniques for low power. Her research interests include design automation techniques for low-power systems, high-level synthesis, and DSP system design. She received the B.Tech degree in electrical engineering from IIT, Kanpur, India, and the MS from the University of California, Berkeley. Mehra is a student member of the IEEE.



**Jan M. Rabaey** is a professor in the Electrical Engineering and Computer Science Department of the University of California, Berkeley. Formerly, he was a research manager at IMEC, Belgium, where he pioneered the development of the Cathedral II synthesis system for digital signal processing. Rabaey's current research interests include the exploration of architectures and algorithms for digital signal processing systems and their interaction. He is also active in various aspects of portable, distributed communication and computation systems, including low-power design, networking, and design applications. Rabaey received EE and PhD degrees in applied sciences from the Katholieke Universiteit Leuven, Belgium. He is a fellow of the IEEE.

Direct questions concerning this article to Paul Landman, Integrated Systems Laboratory, Texas Instruments Inc., 13510 North Central Expressway, MS 446, Dallas, TX 75243; landman@hc.ti.com.