High-Level Power Estimation

Paul Landman
DSP R&D Center
Texas Instruments
Dallas, Texas

Abstract

The growing demand for portable electronic devices has led to an increased emphasis on power consumption within the semiconductor industry. As a result, designers are now encouraged to consider the impact of their decisions not only on speed and area, but also on power throughout the entire design process. In order to evaluate how well a particular design variant meets power constraints, engineers often rely on CAD tools for power estimation. While tools have long existed for analyzing power consumption at the lower levels of abstraction - e.g. SPICE and PowerMill - only recently have efforts been directed towards developing a high-level power estimation capability. This paper surveys the state of the art in high-level power estimation, addressing techniques that operate at the architecture, behavior, instruction, and system levels of abstraction.

1 Introduction

In the early part of this decade, it became clear that power consumption was becoming a problem. In the high performance arena, microprocessors began to appear that consumed tens of watts, and the trend was toward even higher power consumption. This placed stringent demands on packaging and cooling systems, as well as being a major cost and reliability issue. At the same time, the portable consumer electronics market entered a period of rapid growth. For these battery-operated products, there was similar pressure to reduce power consumption and extend battery life. These factors rapidly led to the emergence of low-power design as a key technology for the 90's.

As designers began to place increasing emphasis on power as a figure of merit, it became clear that while there were tools to assist in estimating performance and area, relatively few addressed power. In the last several years, the picture has improved dramatically. Circuit- and gate-level power analysis and estimation tools are now offered by nearly every major EDA vendor.

While the situation is clearly improved, the problem is still far from solved. Both academic and industrial experts have noted that after exploiting the obvious technology and circuit level optimizations, we are left orders of magnitude from where we need to be. These additional reductions must come from optimizations made at the higher abstraction levels - namely, the architecture, algorithm, and system levels.

Unfortunately, EDA vendors offer few solutions to aid in the exploration of the power domain at these levels. Clearly it is not feasible (nor desirable) to specify/synthesize every design alternative down to the gate level. Tools are needed that operate inherently at the architecture level and above. This has been an active area of university research for some time now, and while there may be little to choose from commercially, academic approaches have begun to appear.

This paper will describe the current state of the art in high-level power estimation. We will begin with a proposal for an analysis-based low-power design methodology. The remainder of the paper will be divided into sections covering emerging architecture, behavior, instruction, and system level power estimation techniques that support this methodology. In comparing the work of different researches, quantitative comparisons of accuracy and speed can be misleading. Instead we will endeavor to give a qualitative feel for how the approaches differ, and from this some conclusions about relative performance can be inferred.

2 An analysis-based design methodology

While logic synthesis has gained widespread acceptance among the industrial design community, high-level synthesis has found a foothold in a relatively narrow range of applications (most notably, DSP). One of the reasons for this is that in order to make the problem tractable, most high-level synthesis systems are forced to assume some fixed architectural template that is unlikely to be optimum for all applications. This suggests that a better approach is to rely on the designer to specify system partitionings and architectural configurations, with the primary function of the tools being to provide feedback on the quality of a particular solution.

The result is an analysis-based low-power design methodology. Working top-down, the designer begins at the system level, partitioning the design into off-the-shelf and custom components. Here, the function of the tools is to aid in producing a power budget indicating which parts of the system will likely be the major power consumers. Rough

ISLPED 1996 Monterey CA USA
0-7803-3571-8/96/$5.00 © 1996

power estimates at this stage of design can save a lot of time wasted later on optimizing the wrong part of the system. With an initial partitioning and power budget in hand, the designer can focus on the individual components of the system, which may be realized as software or hardware. Instruction-level power models of programmable processors will be useful here in helping the designer optimize the software portions of the system. Similarly, behavior- and RT-level power estimators will provide a much needed classification of the power efficiency of different algorithms and architectures that might be used for the dedicated hardware. At all these levels, relative accuracy in the power estimates is much more important than absolute accuracy, since what we really want to know is whether one alternative is better than another.

In the final design stages a more traditional flow applies, utilizing software and hardware compilers along with schematic entry, logic synthesis, layout, and place and route tools. The primary function of gate- and circuit-level power analysis and simulation tools would then be to provide back-end verification of power consumption with sign-off accuracy.

As mentioned above, much of the technology to support this low-power design methodology now exists, either commercially or as university research. In the next few sections of this paper we go on to describe the strategies that have been proposed to realize architecture, behavior, instruction, and system level power estimation.

3 Architecture-level power estimation

The lowest level we will consider is the architecture, or register-transfer, level. At this level of abstraction the primitives are functional blocks such as adders, multipliers, controllers, register files, and SRAM's. The difficulty in estimating power at this level stems from the fact that the gate, circuit, and layout level details of the design may not have been specified. Moreover, a floorplan may not be available, making analysis of interconnect and clock distribution networks difficult.

The strategies proposed, thus far, for RT-level power estimation can be divided into two classes: analytical methods and empirical methods.

3.1 Analytical methods

Analytical methods attempt to relate the power consumption of a particular RTL description to fundamental quantities that describe the physical capacitance and activity of a design. Since design complexity is a good first-order measure of physical capacitance we can roughly divide the techniques presented in this section into complexity-based and activity-based models.

3.1.1 Complexity-based models

One strategy relies on the fact that the complexity of a chip architecture can be described roughly in terms of "gate equivalents". Basically, the gate equivalent count of a design specifies the approximate number of reference gates (e.g. 2-input NAND's) that would be required to implement a particular function (e.g. a 16x16 multiplier). This number can be specified in a library database or provided by the user. The power required for each functional block can then be estimated by multiplying the approximate number of gate equivalents by the average power consumed by each gate. An example of this technique is given by the Chip Estimation System (CES) [1], which uses the following expression for average power:

$$P = \sum_{i \in \{\text{fns}\}} GE_i \left(E_{typ} + C_L^i V_{dd}^2 \right) f A_{int}^i \qquad (1)$$

where GE_i is the gate equivalent count for functional block i, E_{typ} is the average energy consumed by an equivalent gate when active, $C_L{}^i$ is the average capacitive load per gate including fan-out and wiring, f is the clock frequency, and $A_{int}{}^i$ is the average percentage of gates switching each clock cycle within functional block i.

One disadvantage of this technique is that all power estimates are based on the energy consumption of a single reference gate. This does not take into account different circuit styles, clocking strategies, or layout techniques. The approximation is particularly inaccurate for specialized blocks such as memories.

Liu and Svensson improved the situation by applying customized estimation techniques to the different design entities: logic, memory, interconnect, and clock [2]. For example, the power consumed by a memory cell array is modeled as:

$$P_{memcell} = \frac{2^k}{2} \left(c_{int} l_{column} + 2^{n-k} C_{tr} \right) V_{dd} V_{swing} f \qquad (2)$$

where 2^k is the number of cells in a row, c_{int} is the wire capacitance per unit length, l_{column} is the memory column length, 2^{n-k} is the number of cells in a column, C_{tr} is the minimum size drain capacitance, and V_{swing} is the bitline voltage swing.

The logic component of power is estimated in a manner conceptually similar to CES. The basic switching energy is based on a three-input AND gate and is calculated from fundamental technology parameters (e.g. minimum gate width, gate length, and oxide thickness). The total chip logic power is estimated (as before) by multiplying the estimated gate equivalent count by the basic gate energy and the activity factor. The activity factor is provided by the user and assumed fixed across the entire chip.

Finally, interconnect length and capacitance is modeled by a derivative of Rent's Rule. The clock capacitance is based on the assumption of an H-tree distribution network.

The advantage of these complexity-based estimation techniques is that they require very little information. Basically, just a few technology parameters, memory sizes, and a count of gate equivalents are needed.

One disadvantage of the complexity-based methods is that they do not model circuit activity accurately. An overall (fixed) activity factor is typically assumed and, in fact, must often be provided by the user. In reality, activity factors will vary with block functionality and with the data being processed. So even if the user provides an activity factor that results in a good estimate of the total chip power, the predicted breakdown of power between modules is likely to be incorrect, making it difficult to perform meaningful architectural trade-offs.

3.1.2 Activity-based models

Activity-based models address this issue to some extent. So far all efforts in this area have focused on using the concept of entropy from information theory as a measure of the average activity in a circuit [3][4]. The basic idea is to try to relate the power that a functional block consumes to the amount of computational work it performs. Entropy is a useful metric from information theory for measuring computational work.

Najm [3], for example, observes that power is proportional to the product of physical capacitance and activity. He then uses area as a measure of physical capacitance and entropy as a measure of activity:

$$P \propto Capacitance \times Activity \propto Area \times Entropy \qquad (3)$$

Leveraging off previous work [5][6], the area of a block's average minimized implementation is related to the number of boolean inputs, n, and to the total entropy of its m outputs, H_o:

$$Area \propto \left\{ \begin{array}{ll} \frac{2^n}{n} H_o & \text{as n} \rightarrow \infty \\ 2^n H_o & \text{for n} \leq 10 \end{array} \right\} \qquad (4)$$

Using the approximation that entropy decreases quadratically with logic depth, he is able to estimate the average entropy of all the nodes in a functional block as a function of the total entropies of its inputs and outputs:

$$Entropy \approx \frac{2/3}{n+m} (H_i + 2H_o) \qquad (5)$$

Najm's power estimation methodology then consists of running an RTL simulation of the design to measure the input and output entropies of the functional blocks and using (3)-(5) to translate these measurements into a prediction of average power. Najm notes that the approach has some significant hurdles to overcome. First, no timing information enters into the above calculations and, therefore, glitching power is not accounted for in any way. Also, there is the implicit assumption in (3) that capacitance is uniformly distributed over all nodes.

Clearly, the accuracy of these techniques is limited; however, they may prove useful for relative architectural comparisons, which as mentioned before is the main function of high-level power estimation tools. Still, these information theoretical approaches are in their infancy and much work needs to be done to demonstrate their value in practice.

All of the analytical power estimation methods described in this section (both complexity- and activity-based) have the advantage of requiring very little information as input. In some sense, this is also a disadvantage in that it is difficult to capture the power attributes of different functional blocks using only parameters such as gate equivalent count or entropy. Therefore, power predictions based on these techniques may not have a strong relation to real hardware.

3.2 Empirical methods

The empirical models discussed in this section offer one possible solution to this problem. Rather than trying to relate the power consumption of RTL components to fundamental parameters, the strategy here is to "measure" the power consumption of existing implementations and produce a model based on those measurements. In other words, these techniques employ a *macromodeling* approach to architectural power estimation.

Clearly, this approach is best suited for designs that will be built using a library-based approach, but this is not a necessity. For example, even if a designer intends to build the functional blocks for his architecture from scratch, it is still likely that models based on previous implementations will give good ballpark power figures. If no previous data is available for a particular block, then analytical models may be more appropriate.

The techniques that fall into the category of empirical methods can further be subdivided into those that assume fixed signal activities and those that account for variations in data and instruction statistics.

3.2.1 Fixed-activity models

The first proposal for a fixed-activity macromodeling strategy was the Power Factor Approximation (or PFA) method [7]. The power consumed by a given architecture is approximated by the expression:

$$P = \sum_{i \in \{\text{all blocks}\}} \kappa_i G_i f_i \qquad (6)$$

where each functional block i is characterized by a PFA proportionality constant κ_i, a measure of hardware complexity G_i, and an activation frequency f_i.

For example, the hardware complexity of a multiplier is related to the square of the input word length, so $G_{mult}=N^2$. The activation frequency is simply the frequency with which multiplies are performed by the algorithm, f_{mult}. Finally, the PFA constant κ_{mult} is extracted empirically

from past multiplier designs (taken from ISSCC proceedings) and shown to be about 15 fW/bit^2-Hz for a 1.2 µm technology at 5V. The resulting power model is:

$$P_{mult} = \kappa_{mult} N^2 f_{mult} \quad (7)$$

Although the authors only explicitly discuss models for multipliers, memories, and I/O drivers, the PFA method can be viewed as a general technique for individually characterizing an entire library of RT-level functional blocks. The power models can be parameterized in terms of whatever complexity parameters are appropriate for that block. For instance, for the memory, the storage capacity in bits is used and for the I/O drivers the word length alone is adequate.

The weakness of fixed-activity models, of course, is that they do not account for the influence that data activity can have on power consumption. For example, the PFA constant κ_{mult} is intended to capture the intrinsic internal activity associated with a multiplier unit; however, since it is taken to be a constant, there is the implicit assumption that the inputs do not affect the multiplier activity, which is not the case. As an example of this phenomenon, Figure 2 (which will be discussed fully in Section 3.2.2) shows how the power consumption of an LMS noise cancellation filter varies for different data streams.

3.2.2 Activity-sensitive models

Activity-sensitive empirical power models have been developed in an attempt to remedy this situation. These models endeavor to account in some way for the influence that data activity statistics can have on power.

A simple example is the RT-level power estimation tool called ESP [8]. Although ESP relies for the most part on a fixed-activity model, it does provide some capability of measuring vector-dependent power. ESP is fundamentally a cycle-based simulator targeted at a RISC processor. As object code is executed, ESP monitors which blocks in the architecture are activated, adding a fixed power contribution for each. The implicit assumption is that the power consumed by each component has been empirically measured prior to simulation. The datapath power model accounts to some extent for input vector activity by using a power model that has a constant portion and a portion that is proportional to the number of bit transitions *n* in the input vector:

$$P = P_{const} + n \cdot P_{change} \quad (8)$$

Another activity-sensitive architectural power analysis tool called SPA has also been developed [9][10][11]. The approach is based on the concept of activity profiling. Specifically, prior to power analysis, an RT-level simulation (e.g. VHDL) of the design in question is carried out for typical instruction and data inputs. During this simulation, the activity of the design entities and signals in the data and control paths are monitored and recorded. These activity statistics are then fed into power models that explicitly

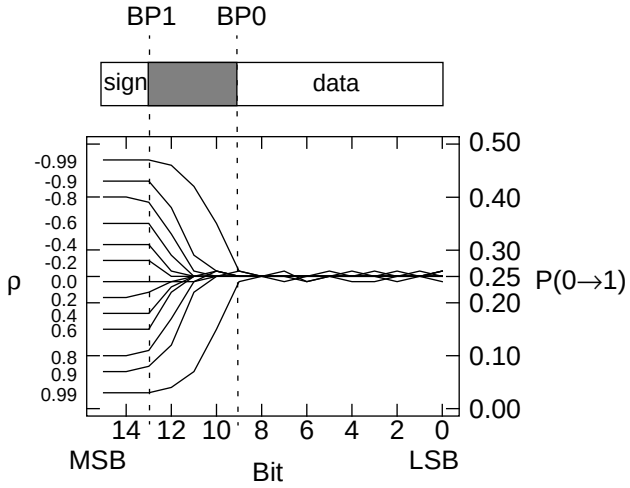

Figure 1. Bit transition activity for data streams with varying temporal correlation, ρ

account for activity as well as complexity. Two different activity models have been described - one for the datapath and one for the control path.

The datapath activity model is referred to as the dual bit type, or DBT, model. It is based on the observation that fixed-point, two's-complement data streams are characterized by two distinct activity regions as shown in Figure 1. The data bits (LSB's) exhibit activity similar to uniformly distributed white noise. The activity of the sign bits (MSB's) depends on the sign transition probability, which is related to the temporal data stream correlation, $\rho = \mathrm{cov}(X_{t-1}, X_t)/\sigma^2$. Different empirically derived coefficients are used to characterize the capacitance switched in the data (C_U) and sign (C_S) regions of various functional blocks:

$$P = (N_U C_U + N_S C_S) V_{dd}^2 f \quad (9)$$

The expression can be extended to more complex multiparameter models using vector notation with arrays of capacitance (**C**) and complexity (**N**) parameters.

In the control path the activity-based control, or ABC, model is used. Unlike datapath words which have a very definite structure, words in the control path often are formed by concatenation of a number of independent fields or boolean flags. Thus, we cannot rely a priori on any particular structure when deriving an activity model for control streams. As a result, the ABC model falls back on more traditional measures of activity: transition probability, α, and signal probability, *P*. Combining these activities with complexity parameters such as the number of inputs (N_I), outputs (N_O), and min-terms (N_M) of a finite state machine (FSM) block, we can derive power models for various controller implementations. The model for an FSM implemented in standard cells, for example, might be:

$$P = (C_I \alpha_I N_I N_M + C_O \alpha_O N_O N_M) V_{dd}^2 f \quad (10)$$

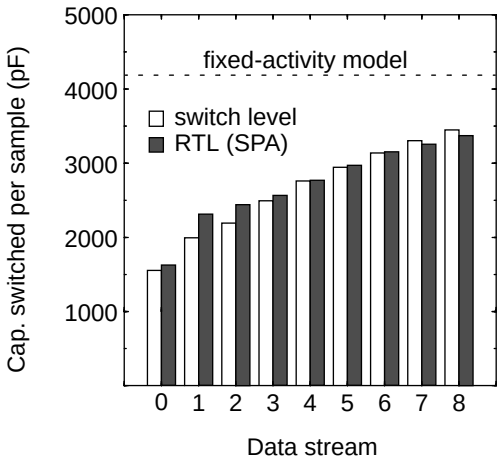

Figure 2. RT-level SPA estimates vs. switch-level simulation of an LMS noise canceller for different data streams

where C_I and C_O are capacitance coefficients empirically extracted from previous standard-cell controller implementations. SPA also includes custom models for interconnect and memory.

A commercial tool called WattWatcher/Architect (offered by Sente, Inc.) relies on techniques similar to those used by SPA, particularly in the area of datapath power modeling [12]. In addition to simulation-based activity profiling, however, it also offers probabilistic activity propagation. It is capable of analyzing a 120,000 gate architecture in 34 minutes.

One of the advantages of empirical models is that they have a strong link to real implementations. Figure 2 compares RT-level predictions from SPA to switch-level simulation of a fully laid out LMS noise canceller. The figure also shows the advantage activity-sensitive can have over fixed-activity techniques.

4 Behavior-level power estimation

As we move up in abstraction level, power estimation becomes even more difficult. Much less is known about a design at the behavior or algorithm level than was known at the architecture level. The typical approach is to assume some architectural style or template and produce power estimates based on it. Some of the numerous unknowns that must be predicted include the foreground/background memory configuration, the number of memory accesses, the bus architecture and average wire length, the number of bus transactions, the control path complexity, and the control line activity.

In studying this list it becomes apparent that some of these parameters relate to the physical capacitance of the resources being accessed, while others describe the activity of those resources. Activity prediction is perhaps the more interesting of the two problems. Behavioral power estimation techniques can be roughly divided into two camps - those that use static activity prediction and those that use dynamic activity prediction.

4.1 Static activity prediction

The access frequency of a resource is important since the more often a resource is activated, the more power it will consume. The object of static activity prediction is to produce an estimate of the access frequency for different hardware resources by analysis of the behavioral description of the function to be implemented. This description could be in the form of a C, Verilog, or VHDL program or it could be represented as a control-data flow graph (CDFG), as is common in high-level synthesis systems. Since only one pass through the program is required, a key advantage of the static profiling approach is its speed.

For programs with no data dependencies, the analysis is quite straightforward and yields the desired access counts for the different operations required by the algorithm. In the more typical case where data-dependent conditionals, branches, and loops are present the situation is more complicated and we must resort either to statistical approximations or dynamic profiling techniques.

Mehra and Chandrakasan have developed a behavioral power estimation strategy using static profiling in the context of the HYPER-LP high-level synthesis system [13][14]. The power required to execute a behavior is expressed as:

$$P = \sum_{r \in \left\{ \begin{array}{c} \text{all datapath, control path,} \\ \text{memory, and bus resources} \end{array} \right\}} f_r C_r V_{dd}^2 \qquad (11)$$

where f_r is the access frequency of resource r as determined by static analysis of the CDFG. The capacitance C_r switched when resource r is activated is determined using empirical fixed-activity models (see Section 3.2.1). For example, the control path model was built by benchmarking the switching capacitance of controllers for 46 different design examples (see Figure 3). From the figure it's clear that while power models at this level of abstraction don't offer a high degree of absolute accuracy, they do capture general trends, which as stated before is the real goal of high-level power estimation.

4.2 Dynamic activity prediction

Dynamic profiling is another technique for determining the activation frequencies of various resources. In this approach, a simulation of the desired behavior is performed for a user-supplied set of inputs. During this simulation, activity statistics are gathered regarding the frequency of various types of operations and memory accesses. These frequencies are then plugged into a model similar to (11) to

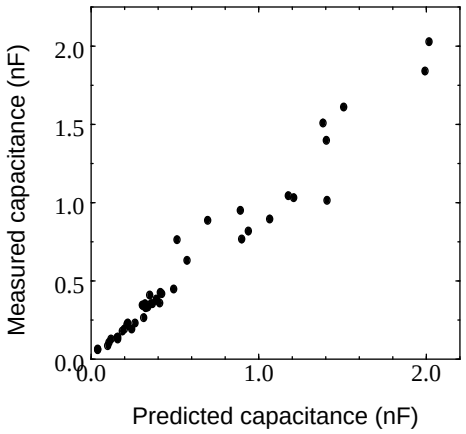

Figure 3. Predicted controller capacitance vs. switch-level simulation [13]

obtain a power estimate. The advantage, of course, is that data dependencies are easily handled. The disadvantages are that it's much slower than the static approach and that it requires the user to supply a set of typical input vectors.

One example of the dynamic approach is the Profile-Driven Synthesis System (PDSS) [15]. The input to the system is a behavioral subset of VHDL. Prior to simulation, the system automatically inserts activity probes. The activation statistics for each datapath operation type are then plugged into what amounts to a library of fixed-activity empirical power models. The controller FSM is assumed to be of the PLA type and empirical power models based on simulations of a randomly generated FSM benchmark set are used to predict its power.

Power-Profiler uses a similar strategy for behavioral power estimation [16]. One key difference is that rather than producing a single average estimate of power consumption, Power-Profiler produces a profile of power versus time. This gives the designer some feel for peak, as well as average, power consumption. It should be noted, however, that an averaging window of one clock cycle is used, which tends to smooth out the power peaks substantially. Therefore, while the tool gives the designer a rough feel for where power peaks might occur, the exact instantaneous amplitude of those peaks is not reported.

5 Instruction-level power estimation

Most behavior-level power estimators assume architectural models corresponding to dedicated hardware implementations. It is also possible to realize a given behavior in software on a programmable instruction set processor. In this case, an instruction-level power model is appropriate.

Tiwari *et al* proposed just such a model for embedded general-purpose and DSP processors [17]. The strategy they describe is most similar to the empirical macromodeling approach of Section 3.2. Each available instruction is placed in a loop and executed on the target processor. During this process current measurements are taken, and the average current drawn by each instruction is stored in a table of *base costs*.

The model also handles what are referred to as inter-instruction effects. In a real program, the change of circuit state between two instructions leads to current consumption that is higher than predicted by the base cost. Therefore, an additional fixed circuit-state overhead current must be added to the base cost for each instruction. The magnitude of this correction factor can be determined by executing pairs of instructions while measuring current. Additional effects such as pipeline stalls and cache misses are also considered in the model.

To date, the model has been used to characterize the power consumption of the Intel 486DX2, the Fujitsu SPARClite 934, and a Fujitsu embedded DSP processor. The authors note that while accurate for most instructions, the estimates produced for the DSP MAC instruction can err significantly. In fact, while the base cost for a packed MAC:LAB instruction is 36.9 mA, the overhead cost can vary from as little as 1.4 mA to as much as 33.0 mA. The variation is caused by the impact of operand activity on the multiplier. In order to account for this, an activity-sensitive model similar to those described in Section 3.2.2 would be needed.

6 System-level power estimation

At the earliest stages of design specification we can consider performing system-level power estimation. Here the goal is to come up with a rough power budget accounting for all the components in a system. This should include the analog, digital, mixed signal, and even electromechanical portions of a system. A power exploration tool at this level of abstraction would be quite useful for identifying power bottlenecks before any time is wasted optimizing the wrong part of the system. It would also be helpful in determining the best way to partition the desired functionality into individual components. The partitioning and level of integration of a system can have a profound effect on the overall power consumption.

A tool called PowerPlay has been recently developed that encompasses these capabilities [18]. PowerPlay is available as a world wide web application and employs a spreadsheet-like interface to facilitate hierarchical design entry and rapid exploration of design partitionings and parameters variations (e.g. supply voltage and clock frequency).

The power models used by PowerPlay leverage off the existing high-level power estimation literature and currently are primarily empirical in nature. The models are placed in a hardware library and are shared among users. New mod-

els can be created easily using user-defined parameterized expressions or values taken straight from product data sheets. The PowerPlay framework has been used successfully to model the power of the InfoPad system, a portable multimedia terminal [19]. This illustrates PowerPlay's ability to model programmable processors, ASIC's, memories, FPGA's, radio modems, and LCD displays.

7 Conclusions

The development of tools to support a low-power design methodology has been an area of active research for the last several years. While much of the literature deals with circuit- and gate-level techniques, a significant amount has also been published on high-level power estimation. This paper has made an attempt to gather the applicable research into one place and describe how the individual pieces can be integrated to form a coherent whole.

Power estimation strategies are now available that operate at the architecture, behavior, instruction, and even system levels of abstraction. While the majority are from academic circles, some tools have also begun to appear commercially, and many more are sure to be offered in the near future.

Even so, the field of high-level power is in its infancy. Much remains to be done in order to demonstrate the robustness and applicability of the techniques in a realistic industrial setting. It is clear, however, that high-level analysis tools fill a significant gap in current low-power design methodologies, allowing designers to make more informed decisions from the earliest stages of system implementation.

Acknowledgments

The author would like to thank Jan Rabaey for his help during the preparation of this manuscript. Some of the research described in this paper was funded by ARPA grant J-FBI 93-153 and by a fellowship from the National Science Foundation.

References

[1] K. Müller-Glaser, K. Kirsch, and K. Neusinger, "Estimating Essential Design Characteristics to Support Project Planning for ASIC Design Management," *IEEE International Conference on Computer-Aided Design '91*, Los Alamitos, CA, pp. 148-151, November 1991.

[2] D. Liu and C. Svensson, "Power Consumption Estimation in CMOS VLSI Chips," *IEEE Journal of Solic-State Ciruicts*, pp. 663-670, June 1994.

[3] F. Najm, "Towards a High-Level Power Estimation Capability," *1995 International Symposium on Low-Power Design*, pp. 87-92, April 1995.

[4] D. Marculescu, R. Marculescu, and M. Pedram, "Information Theoretic Measures of Energy Consumption at Register Transfer Level," *1995 International Symposium on Low-Power Design*, pp. 81-86, April 1995.

[5] N. Pippenger, "Information Theory and the Complexity of Boolean Functions," *Mathematical Systems Theory*, vol. 10, pp. 129-167, 1977.

[6] K-T Cheng and V. Agrawal, "An Entropy Measure for the Complexity of Multi-Output Boolean Functions," *27th ACM/IEEE Design Automation Conference*, pp. 302-305, June 1990.

[7] S. Powell and P. Chau, "Estimating Power Dissipation of VLSI Signal Processing Chips: The PFA Technique," *VLSI Signal Processing IV*, pp. 250-259, 1990.

[8] T. Sato, Y. Ootaguro, M. Nagamatsu, and H. Tago, "Evaluation of Architecture-Level Power Estimation for CMOS RISC Processors," *1995 Symposium on Low-Power Electronics*, pp. 44-45, October 1995.

[9] P. Landman, *Low-Power Architectural Design Methodologies*, Ph.D. Dissertation, UC Berkeley, August 1994.

[10] P. Landman and J. Rabaey, "Architectural Power Analysis: The Dual Bit Type Method," *IEEE Transactions on VLSI Systems*, pp. 173-187, June 1995.

[11] P. Landman and J. Rabaey, "Activity-Sensitive Architectural Power Analysis," *IEEE Transactions on CAD*, June 1996.

[12] WattWatcher Product Sheet, Sente Corp., Chelmsford, MA.

[13] R. Mehra, "High-Level Power Estimation and Exploration," *1994 International Workshop on Low Power Design*, pp. 197-202, April 1994.

[14] A. Chandrakasan, M. Potkonjak, J. Rabaey, and R. Brodersen, "Optimizing Power Using Transformations," *IEEE Transactions on Computer-Aided Design*, pp. 12-31, January 1995.

[15] N. Kumar, S. Katkoori, L. Rader, and R. Vemuri, "Profile-Driven Behavioral Synthesis for Low-Power VLSI Systems," *IEEE Design & Test of Computers*, pp. 70-84, Fall 1995.

[16] R. San Martin and J. Knight, "Optimizing Power in ASIC Behavioral Synthesis," *IEEE Design & Test of Computers*, pp. 58-70, Summer 1996.

[17] V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization," *IEEE Transactions on VLSI Systems*, pp. 437-445, December 1994.

[18] D. Lidsky and J. Rabaey, "Early Power Exploration: A World Wide Web Application," accepted to *33rd Design Automation Conference*, Las Vegas, 1996.

[19] S. Sheng, A. Chandrakasan, and R. Brodersen, "A Portable Multimedia Terminal," *IEEE Communications Magazine*, pp. 64-75, December 1992.