Tools and Methodologies for Low Power Design

Jerry Frenkil
Sente, Inc.
3 Summer St., Chelmsford, MA 01824
jfrenkil@powereda.com

Abstract -- Designing for low power has become increasingly important in a wide variety of applications, including wireless telephony, mobile computing, high performance computing, and high speed networking. Despite reductions in power supply voltages, power consumption continues to rise and demands increased support from EDA tools and methodologies. Various tools have emerged to address different levels of the power problem, yet conventional methodologies often focus on the low leverage aspects. This paper will survey existing commercial tools used in low power design and present them in the context of an architecture focused low power design methodology.

I. INTRODUCTION

The issue of power consumption in integrated circuit design continues to grow. Once a concern only to designers of very low power electronics, power consumption now is an issue for designers of all types of integrated circuits. Especially feeling the pain are designers of microprocessors, multi-media and digital signal processors, high-speed networking devices, and of course battery powered and wireless electronics.

Market forces are demanding lower power for many reasons, some obvious and others subtle. Battery life is a well understood advantage of lower power consumption, however reliability, performance, cost, and time to market all benefit from lower power. At the same time however, technological forces are pushing power consumption to higher and higher levels.

Design Automation Conference (R)
 0-89791-847-9/97/0006/$3.50 DAC 97 - 06/97 Anaheim, CA, USA

The initial response to the problem was to lower the power supply voltage and this trend continues. However, it has become clear that while this is perhaps a necessary part of the solution, it is by no means the full solution, as power consumption has continued to rise even while power supply voltages have been dropping. Accordingly, researchers and designers have in recent years been focusing on design tools and methodologies as a more leveraged approach to reducing power consumption. This paper will describe currently available EDA tools for low power design and how they can be incorporated into a design flow focused on achieving minimum power consumption. The paper will begin with a brief review of the different types and sources of power, followed by descriptions and examples of representative design tools. A methodology for low power design will be constructed out of these design tools illustrating the state of the art in low power design. Finally, the paper will conclude with a brief description of future work in the area.

II. POWER ISSUES

Minimizing power consumption is not simply an altruistic activity - all else being equal, a device consuming less power will accrue several desirable advantages which are otherwise difficult or impossible to obtain. An obvious advantage is longer battery life for wireless devices.

Somewhat less obvious advantages are increased reliability and performance, both of which are temperature dependent, as shown in Figure 1. Long term reliability (MTBF) drops by 50% for each 10 degree rise in junction temperature [1]. In addition, transistor currents, and hence performance, drop by approximately 3% for each 10 degree rise [2], providing significant motivation to minimize power, which in turn minimizes junction temperature.

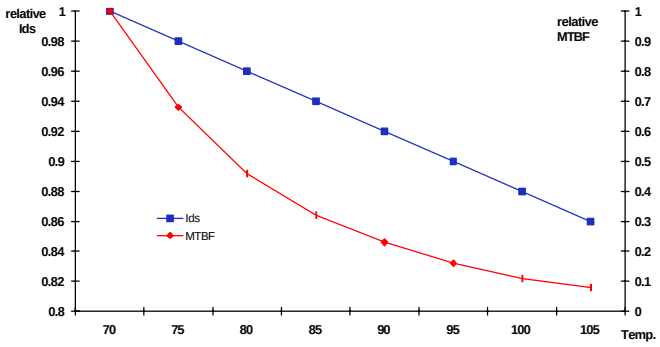

Figure 1: Temperature Effects

Cost is another key effect of power. Elevated levels of power consumption require more sophisticated, and more costly, packaging solutions to maintain a desired junction temperature. This affects the integrated circuit packaging as well as system packaging: increased air flow requirements and larger fans. Simply adding a heat sink to a chip can increase its unit cost substantially, directly affecting the project's, and company's, bottom line.

To address these issues directly, it is essential to understand the different types and sources of power.

A. Power Sources

An IC's total power consumption is comprised of two different types, static and dynamic. The primary distinguishing factor between the two is that dynamic power is frequency dependent while static is not. Static power is defined to be the product of the power supply voltage and a static current, which itself has two components: leakage current and through current. Leakage currents, which occur in all MOS devices, are due to sub-threshold transistor operation and reverse bias diode leakage. Through currents also occur in normal operation but are due to transistors being continuously operated in their saturation (always on) regions and arise out of design decisions to employ analog techniques or resistive pull-ups. The magnitude of through currents is usually in the micro-amp to milli-amp range. Leakage currents, by comparison, are parasitic effects which are orders of magnitude smaller and are usually ignored, except in those devices that have extremely long standby times or operate at extremely low voltages.

Dynamic power also has a couple of components, crowbar power and (capacitive) load power. Crowbar power represents the crowbar currents that flow from power to ground when the transistor stacks switch state, while the capacitive load power represents the currents required to charge the fanout load capacitance. In an ASIC, or library based, design methodology these dynamic currents are combined into *cell* and *load* currents, where the cell currents include the crowbar currents as well as the currents required to charge the parasitic capacitances *internal* to the cell, while the load currents represent the current required to charge the load capacitance's external to the cell.

These currents are shown in Figure 2, which illustrates the flow of each of these currents. The load current, Iload, charges the load capacitance. Isc illustrates the current which flows from the power supply to ground which represents the crowbar current in a fully complementary CMOS circuit, or a through current in an analog circuit. Ileakage represents the leakage currents which unavoidably flow through the reverse-biased parasitic diodes.

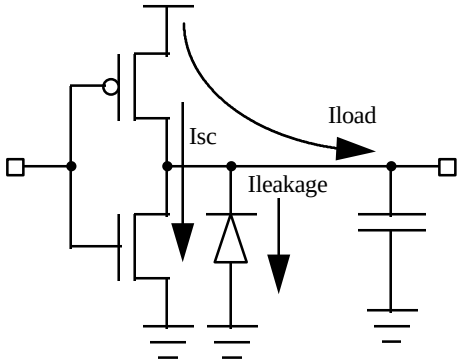

Figure 2: Power Sources

B. Power Types

In addition to understanding the sources of power consumption, it is also essential to understand the measurement types of power consumption, as each type is used for a different purpose. The different types are time-averaged, maximum, RMS, and (instantaneous) peak.

Time-averaged power, also simply known as average power, represents a single value for power consumption, obtained by integrating power on a cycle by cycle basis over a number of clock cycles. Average power is used to calculate battery life and junction temperature, and is thus the most widely used type of power calculation. Maximum power figures are used to size power busses to

satisfy IR drop requirements in order to meet worst case noise margins.

RMS power is also used for sizing, however these values are used to meet electromigration guidelines. Electromigration rules are usually sensitive to both steady state and periodic current flow, so RMS or some form of modified RMS calculations are used to verify that the current density rules are not violated.

Lastly, instantaneous peak currents, sometimes referred to as di/dt currents, are used to determine the allowable parasitic inductance presented to the device. These values and calculations are essential for minimizing ground bounce effects, both on the chip's IOs as well as on the internal logic.

With this understanding of the physical sources and effects of power, along with a classification of the measurement types, the various design tools for low power design can be evaluated for their applicability to a low power design flow.

III. LOW POWER DESIGN TOOLS

Numerous EDA tools exist for use in minimizing power consumption. Some are targeted specifically for use in the power domain, while others are more general purpose and may be typically used for other tasks, such as timing analysis. In this survey of currently available commercial EDA tools, each tool is classified according to the abstraction layer upon which it operates. Four layers are chosen for classification: transistor (or switch), logic (or gate), architecture (or RTL), and behavior (or system). Note that due to space limitations only the primary examples of these tools are described for each abstraction layer.

A. Transistor Level Tools

Transistor level analysis tools are generally very accurate and mature. Examples of these types of tools are SPICE and its many variants, Epic's PowerMill [3], and Avanti's ADM [4].

The primary advantages of transistor level tools are their accuracy and well-accepted abstraction - most IC designers understand transistor level analysis and rely upon them. However, these tools do have significant issues in their applicability to low power design: capacity and run time characteristics limit their use to small circuits, or very limited depths of simulation vectors for larger circuits.

These tools are utilized primarily in two different use models. The first is to characterize circuit elements in order to create timing and power accurate models for use at the higher abstraction layers. The second is to verify, with the highest levels of accuracy, that the completed transistor level design meets the previously targeted power specification.

An optimization tool operating at the transistor level is Epic's AMPS [5] which is used to automatically size transistors to trade-off timing margin for lower power. It is used in custom design flows that focus on transistor level design.

Other transistor level tools of note are Epic's RailMill [6] and Simplex's Thunder & Lightning [7] which are used to analyze voltage drop and electromigration effects on transistor level layouts.

B. Logic Level Tools

Numerous logic level power analysis tools are currently available from a several vendors: Mentor's QuickPower [8], Sente's WattWatcher/Gate [9], and Synopsys' DesignPower [10], among others. Each of these tools operates on a gate level netlist and assumes the availability of a gate level power library. Such a library contains power models for each individual cell, such as inverters, nand gates, and flip-flops. The model for each cell describes parameters that affect power consumption such as pin capacitances and the energy consumed by the cell when an energy consuming event occurs. Static power due to both leakage and through currents can also be described. The format and specifics of which data types can be represented varies from simulator to simulator, since at the present time no standard has yet emerged for power modeling.

These tools operate by computing power on a cell by cell basis, combining nodal activities obtained from a logical simulation of the netlist with cell specific power data from a power library. Some of these tools operate off of a vcd (value change dump) file, while others collect data activities from simulator PLI routines. DesignPower also has a probabilistic mode in which signal activities are computed probabilistically in lieu of using simulated data.

Gate level power analysis tools are currently in their first generation, with second generation tools in planning or development. Second generation tools can be characterized by the utilization of standard power

libraries, better state-dependency handling, richer macro-cell modeling capabilities, and more detailed handling of static power conditions.

Although less mature than transistor level analysis, gate level power analysis is generally well understood. Compared to the transistor level tools, gate level tools trade off accuracy for significant improvements in run time and capacity. They also tend to fit into ASIC based design flows much better than transistor level tools, as ASIC vendors generally do not make transistor netlists available. However, gate level tools are still limited in overall capacity, and are generally used more in a verification role since the design must be completed, synthesized, and simulated before meaningful power results can be obtained.

PowerCompiler from Synopsys is a gate level power optimization tool. It uses several optimization techniques to achieve the lowest power under timing constraints, and is claimed to produce an average power reduction of 10% to 15% on synthesizable logic.[11]

C. Architecture Level Tools

The architectural level, or register transfer level (RTL), is the design entry point for most digital designs today, and recent research has demonstrated that design decisions made at this level can have a dramatic impact on the design's overall power characteristics. Thus the availability of efficient and accurate tools at this level of abstraction is of the utmost importance. [12]

Most of the work in this area to date has been academically or research oriented [13, 14] however Sente's WattWatcher/Architect is the first commercially available tool operating at the architectural level. WattWatcher/Architect reads Verilog or VHDL RTL descriptions and utilizing simulation data from an RTL simulation computes a power estimate for the entire chip, including memories, clocks, IOs, datapath and control logic. Power estimates are linked to lines of source code to facilitate an understanding of what constructs result in how much power.[15]

Similar to the comparison between transistor level tools and gate level tools, architectural power estimation trades off accuracy for even larger improvements in run time and capacity. Reasonable accuracy (WattWatcher/Architect claims to be within 20% to 25% of silicon) is maintained while execution speed and capacity are significantly enhanced, thereby enabling design space exploration which would be too slow or impossible to do at the gate level.

Architectural level tools can be used to quickly gain an understanding of which modules consume the largest amounts of power. Once this is understood, a detailed view of the frequency domain usually yields insights into the frequency hot-spots and power consumption sources. The architectural level is also the best level at which to evaluate the effects on power consumption of clock gating strategies, which are especially important in reducing average power. The effects of application software on power consumption are best observed at this level as well, due to the run time efficiencies of simulating at the architectural level.

D. Behavior Level Tools

Generally the least explored of the abstraction layers in terms of power, the behavior level is currently unsupported by commercial tools. Nonetheless, it is an active area of academic research, with results reported for Hyper-LP[16], Explore[17], and PowerPlay[18].

IV. A LOW POWER DESIGN METHODOLOGY

To fully manage and optimize power consumption, a design methodology must address power consumption issues at each stage of the design process and at each level of design abstraction. Target power specifications must be developed at the very beginning and the design should be checked against these specifications at each abstraction level. The design can be optimized, either manually or automatically, at each level with the most effective optimizations available at the higher levels of abstraction.

A. Power Management

Power management can be defined to be the control of power consumption across the entire design over all abstraction levels. It differs from power optimization, which is the minimization of power at a given abstraction level. Thus to achieve a truly power efficient design, not only must one *optimize* power at each level, but one must also *manage* power consumption over the entire design. For example, one common power management technique is to increase the size of an on-chip cache to reduce the number of off-chip accesses. From one perspective, this appears to increase power, because a larger memory will consume more power per access than a smaller one. However, this technique actually reduces the total amount of power consumed because the increase in cache power consumption is more than counter balanced by the savings incurred from fewer off-chip accesses.

B. Feed Forward Design

Designing for low power, as described above, entails starting with a target specification, designing an architecture to meet that specification, performing abstraction level specific optimizations to minimize the power, and then checking to verify that the design is on track to meet the initial specification. This methodology features a feed-forward approach to design, as opposed to the more conventional feed-back methodology. Most design projects have historically utilized feed-back design methodologies, wherein a target architecture or concept is implemented, via gate level synthesis or even layout, in order to evaluate its power characteristics. Illustrated in Figure 3, it features a relatively lengthy feedback loop from the analysis results at the gate or transistor level.

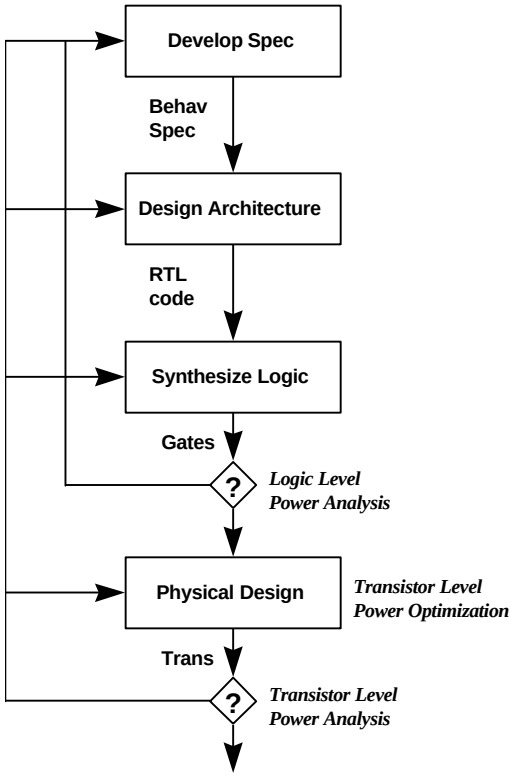

Figure 3: A Feed-Back Design Methodology

Thus information about the design's power characteristics are not obtained until quite late in the design process. Once this information is available, it is *fed-back* to the higher abstraction levels to be used in determining what to do to minimize the power. The farther the power analysis results are in excess of the target specification the higher in the abstraction levels one must return in order to change the design to try and meet the spec.

In the *feed-forward* approach, illustrated in Figure 4, these lengthy, cross synthesis, cross-abstraction feed-back loops are replaced with more efficient abstraction specific loops. Thus the design that is fed forward to the lower abstraction levels is much less likely to be fed back for reworking, and the analysis performed at the lower levels becomes less of a *design* effort and more of a *verification* effort. For example, architecture level power analysis is used to ensure that the RTL design passed forward to synthesis is a design that is capable of meeting the target power specifications. Other information that would be fed-forward would include floor-plans, clock distribution styles, module micro-architectures, critical net names (for power or timing), and power bus topologies. Later in the design flow, gate level and transistor level power analysis is targeted at verifying that the design still meets the power specifications and that any lower level optimizations have had the desired result.

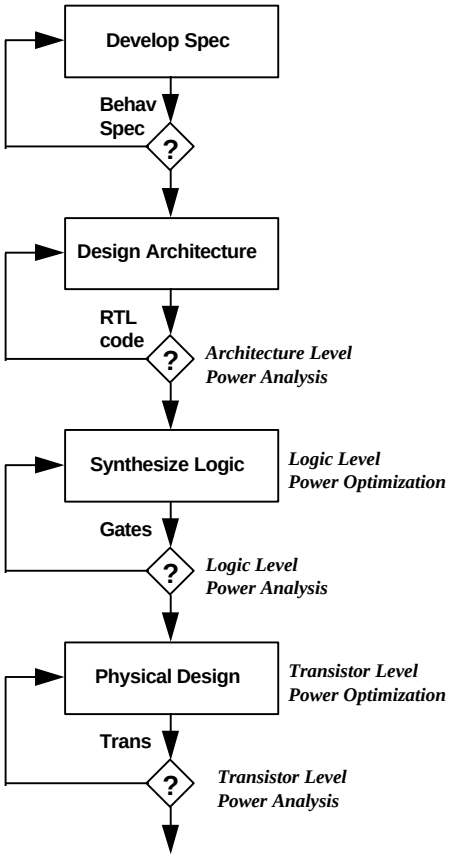

Figure 4: A Feed-Forward Design Methodology

Furthermore, this approach encourages design efforts to be spent up front, at the architectural level, which is where research has indicated orders of magnitude savings in power exist. Conversely, lower level gate and circuit level optimizations typically offer improvement by a factor of two or less.[19]

C. Expectations

It is often unclear what to expect when designing with new constraints, such as power, or when adopting a new methodology. Setting reasonable expectations for schedules and other project metrics can go a long way towards ensuring a smoothly run project.

In general, it should be expected that the accuracy of higher level tools will be, to some degree, less than the lower level tools but will be counterbalanced by significant run time and capacity improvements. The feed forward methodology is designed to take advantage of these execution time improvements so that the major design decisions for power, as well as other parameters, should be made at the highest levels of abstraction. The feed forward methodology is also designed to utilize the slowest tools as little as possible, typically only as a verification activity near the end of the implementation phase at the lower abstraction levels.

V. FUTURE WORK

As designs become even larger and more complex, design reuse will become much more prevalent. This will lead to a requirement for efficient, high level power models. Furthermore, these models must be capable of being utilized at the various abstraction levels, since it will not be feasible to verify such mega-chips at the transistor level in reasonable amounts of time. In addition to designing power efficient hardware, application software in many cases will also be judged in terms of power efficiency. Here again, the availability of efficient, high level power models, coupled with high level power estimation tools will be essential.

VI. CONCLUSIONS

Low power design is no longer of interest only to those developing battery powered electronics, but has also become important in a wide variety of applications. While voltage reduction is perhaps the first technique to be employed, it has quickly become obvious that it is insufficient to meet overall power consumption goals. Various power analysis tools have emerged to address these issues, and have motivated new thinking regarding design methodologies. Architectural level design practices and feed forward design methodologies have been developed to utilize these new tools in a comprehensive, top-down methodology for low power design.

REFERENCES

[1] Intel Components and Reliability Handbook, pg. 7-1, 1991/1992.

[2] L. Glasser, D. Dobberpuhl, "The Design and Analysis of VLSI Circuits", Reading, Massachusetts, Addison-Wesley, 1985, pp. 105-106.

[3] Epic Design Technology's PowerMill, http://www.epic.com/products.html

[4] Avanti's ADM, http://www.anagraminc.com/products/index.html

[5] Epic Design Technology's AMPS, http://www.epic.com/products.html

[6] Epic Design Technology's RailMill, http://www.epic.com/products.html

[7] Simplex's Thunder & Ligthning, http://www.simplex.com/ng/products/tnl

[8] Mentor's QuickPower, http://www.mentorg.com

[9] Sente's WattWatcher/Gate, http://www.powereda.com/wwinfo.htm

[10] Synopys's DesignPower http://www.synopsys.com/products/power/power_ds.html#2

[11] Synopsys' Power Compiler, http://www.synopsys.com/products/power/power_ds.html#3

[12] D. Singh, J. Rabaey, M. Pedram, F. Catthoor, S. Rajgopal, N. Sehgal, T. Mozdzen, "Power-conscious CAD tools and methodologies: a perspective," in Proceedings of the IEEE, pp. 570-594, April 1995.

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

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

[15] Sente's WattWatcher/Architect, http://www.powereda.com/wwinfo.htm

[16] A. Chandrakasan, M. Potkonjak, R. M ehra, J. Rabaey, and R. Brodersen, "Optimizing Power Using Transformations", IEEE Transactions on CAD of Integrated Circuits and Systems, pp. 12-31, January 1995.

[17] R. Mehra, J. Rabaey, "Behavioral Level Power Estimation and Exploration", Proceedings of the 1994 International Workshop on Low Power Design, pp. 197-202.

[18] D. Lidsky, J. Rabaey, "Early Power Exploration - A World Wide Web Application", Proceedings of the 1996 Design Automation Conference, pp. 27 - 32.

[19] P. Landman, R. Mehra, J. Ra baey, "An Integrated CAD Environment for Low-Power Design", IEEE Design and Test of Computers, pp. 72-82, Summer1996.