A Survey of Optimization Techniques Targeting Low Power VLSI Circuits

Srinivas Devadas
Massachusetts Institute of Technology
Department of EECS

Sharad Malik
Princeton University
Department of EE

Abstract—We survey state-of-the-art optimization methods that target low power dissipation in VLSI circuits. Optimizations at the circuit, logic, architectural and system levels are considered.

Keywords—low power, optimization, synthesis

I. Introduction

Power dissipation has emerged as an important design parameter in the design of microelectronic circuits, especially in portable computing and personal communication applications. In this paper, we survey state-of-the-art optimization methods that target low power dissipation in VLSI circuits. Optimizations at the circuit, logic, architectural and system levels are considered.

Sources of power dissipation in CMOS devices are summarized by the following expression:

$$P = \frac{1}{2} \cdot C \cdot V_{DD}^2 \cdot f \cdot N + Q_{SC} \cdot V_{DD} \cdot f \cdot N + I_{leak} \cdot V_{DD} \quad (1)$$

where P denotes the total power, V_{DD} is the supply voltage, and f is the frequency of operation.

The first term represents the power required to charge and discharge circuit nodes. Node capacitances are represented by C. The factor N is the switching activity, i.e., the number of gate output transitions per clock cycle.

The second term in Eqn. 1 represents power dissipation during output transitions due to current flowing from the supply to ground. This current is often called short-circuit current. The factor Q_{SC} represents the quantity of charge carried by the short-circuit current per transition.

The third term in Eqn. 1 represents static power dissipation due to leakage current I_{leak}. Device source and drain diffusions from parasitic diodes with bulk regions. Reverse bias currents in these diodes dissipate power. Subthreshold transistor currents also dissipate power. In the sequel, we will refer to the three terms above as switching activity power, short-circuit power and leakage current power.

Most of the optimizations described in the following sections concentrate on minimizing switching activity power at various levels of abstraction. In VLSI circuits that use well-designed logic-gates, switching activity power accounts for over 90% of the total power dissipation [8].

II. Circuit Level

We survey optimizations that reduce switching activity power of individual logic-gates and transistor-level combinational circuits in this section.

32nd ACM/IEEE Design Automation Conference ®

A. Complex Gate Design

In the design of complex gates, e.g., $f = \overline{(a+b) \cdot c}$, choices regarding the placement of individual transistors in the gate can be made. For example, in the N part of the CMOS gate implementing the above function f, the parallel transistor pair $a + b$ can be connected to the gate output or the transistor driven by c can be connected to the gate output. Similarly, given $g = \overline{a \cdot b \cdot c}$, any serial ordering of a, b and c can be chosen in the N part of a CMOS gate implementing g.

It is well known that late arriving signals should be placed closer to the output to minimize gate propagation delay. However, the average power dissipated is dependent on the transition probabilities of the gate inputs and the internal node capacitances. (Internal node capacitance is due to parasitic drain and source capacitance and interconnect capacitance.)

Ordering of gate inputs will affect both power and delay. In [32] and [42] methods to optimize the power and/or delay of logic-gates based on transistor reordering are given. Moderate improvements in power and delay can be obtained by a judicious ordering of transistors within individual complex gates.

B. Transistor Sizing

Transistor sizing in a combinational gate circuit can have significant impact on circuit delay and power dissipation. If the transistors in a given gate are increased in size, then the delay of the gate decreases, however, power dissipated in the gate increases. Further, the delay of the fanin gates increases because of increased load capacitance.

Given a delay constraint, finding an appropriate sizing of transistors that minimizes power dissipation is a computationally difficult problem. A typical approach to the problem is to compute the slack at each gate in the circuit, where the slack of a gate corresponds to how much the gate can be slowed down without affecting the critical delay of the circuit. Subcircuits with slacks greater than zero are processed, and the sizes of the transistors reduced until the slack becomes zero, or the transistors are all minimum size. Variants of the above approach are presented in [42] and [3].

III. Logic Level

We survey optimizations that reduce switching activity power of logic-level combinational and sequential circuits in this section.

A. Combinational

Combinational logic optimization has traditionally been decomposed into two phases: technology-independent optimization and technology-dependent optimization. In the first phase logic equations are manipulated to reduce area, delay or power dissipation. In the second phase the equations are mapped to a particular technology library using technology mapping algorithms, again optimizing for area, delay or power. For a comprehensive treatment of combinational logic synthesis methods targeting area and delay, see [13]. In this section we will survey recently proposed methods to optimize combinational circuits for low power dissipation.

A.1 Don't-care Optimization

Any gate in a combinational circuit has an associated controllability and observability don't-care set. The controllability don't-care set corresponds to the input combinations that never occur at the gate inputs. The observability don't-care set corresponds to collections of input combinations that produce the same values at the circuit outputs. Methods to reduce circuit area and improve delay exploiting don't-care sets have been presented (e.g., [37]). The power dissipation of a gate is dependent on the probability of the gate evaluating to a 1 or a 0. This probability can be changed by utilizing the don't-care sets. A method of don't-care optimization to reduce switching activity and therefore power dissipation was presented in [38]. This method was improved upon in [19] where the effect of don't-care optimization of a particular gate on the gates in its transitive fanout is considered.

A.2 Path Balancing

Spurious transitions account for between 10% and 40% of the switching activity power in typical combinational logic circuits [16]. In order to reduce spurious switching activity, the delays of paths that converge at each gate in the circuit should be roughly equal. By selectively adding unit-delay buffers to the inputs of gates in a circuit, the delays of all paths in the circuit can be made equal. This addition will not increase the critical delay of the circuit, and will effectively eliminate spurious transitions. However, the addition of buffers increases capacitance which may offset the reduction in switching activity.

Methods to reduce rather than completely eliminate spurious switching activity, while adding a minimal number of unit-delay buffers have been proposed. The design of a multiplier with transition reduction circuitry that accomplishes glitch reduction by path balancing is described in [25].

A.3 Factorization

A primary means of technology-independent optimization is the factoring of logical expressions. For example, the expression $a \cdot c + a \cdot d + b \cdot c + b \cdot d$ can be factored into $(a + b) \cdot (c + d)$ reducing transistor count considerably. Common subexpressions can be found across multiple functions and reused. Kernel extraction is a commonly used algorithm to perform multilevel logic optimization for area [5]. In this algorithm, the kernels of the given expressions are generated and kernels that maximally reduce literal count are selected. When targeting power dissipation, the cost function is not literal count but switching activity. Modified kernel extraction methods that target switching activity power are described in [35].

B. Technology Mapping

Once optimized logic equations have been obtained, the task remains to map the equations into a target library that contains optimized logic-gates in the chosen technology. A typical library will contain hundreds of gates with different transistor sizes. Modern technology mapping methods use a graph covering formulation, originally presented in [20], to target area and delay cost functions.

The graph covering formulation of [20] has been extended to the power cost function. Under the zero delay model, the optimal mapping of a tree can be determined in polynomial time, by extending the algorithm of [20]. Various approaches to technology mapping that assume different delay models and target minimal power dissipation have been described [43] [48] [26].

C. Sequential

We survey methods to optimize sequential circuits for low power in this section. Sequential logic optimization methods work at two levels of abstraction; 1) at the State Transition Graph level and 2) at the logic-gate and flip-flop level.

C.1 Encoding

State encoding for minimal area is a well-researched problem [2]. These methods have to be modified to target a power cost function, namely, weighted switching activity. Intuitively, if a state s has a large number of transitions to state q, then the two states should be given uni-distant codes, so as to minimize switching activity at the flip-flop outputs. However, the complexity of the combinational logic resulting from a state assignment should not be ignored. Methods to encode State Transition Graphs to produce two-level and multilevel implementations with minimal power are described in [35] and [47]. A method to re-encode logic-level sequential circuits to minimize power dissipation is presented in [18].

Encoding to reduce switching activity in datapath logic has also been the subject of attention. A method to minimize the switching on buses is proposed in [39]. In this technique, an extra line E is added to the bus which signifies if the value being transferred is the true value or needs to be bitwise complemented upon receipt. Depending on the value transferred in the previous cycle, a decision is made to either transfer the true current value or the complemented current value, so as to minimize the number of transitions on the bus lines. For example, if the previous value transferred was 0000, and the current value is 1011, then the value 0100 is transferred instead, and the line E is asserted to signify that the value 0100 has to be complemented at the other end. Other methods of bus coding are also proposed in [39].

Methods to implement arithmetic units other than in standard two's complement arithmetic are also being investigated. A method of one-hot residue coding to minimize switching activity of arithmetic logic is presented in [11].

C.2 Retiming

Retiming [24] is a well-known optimization method that repositions the flip-flops in a synchronous sequential circuit so as to minimize the required clock period. Polynomial-time algorithms for minimum-delay retiming and minimum-register retiming have been developed.

It has been observed that the switching activity at flip-flop outputs in a synchronous sequential circuit can be significantly less than the activity at the flip-flop inputs. This is because there may be many spurious transitions at the inputs to the flip-flops which are filtered out by the clock. A retiming method that exploits the above observation and targets the power dissipation of a sequential circuit is described in [29].

C.3 Gated Clocks

Large VLSI circuits such as processors contain register files, arithmetic units and control logic. The register file is typically not accessed in each clock cycle. Similarly, in an arbitrary sequential circuit, the values of particular registers need not be updated in every clock cycle. If simple conditions that determine the inaction of particular registers can be determined, then power reduction can be obtained by gating the clocks of these registers [9]. When these conditions are satisfied, the switching activity within the registers is reduced to negligible levels.

The same method can be applied to "turn off" or "power down" arithmetic units when these units are not in use in a particular clock

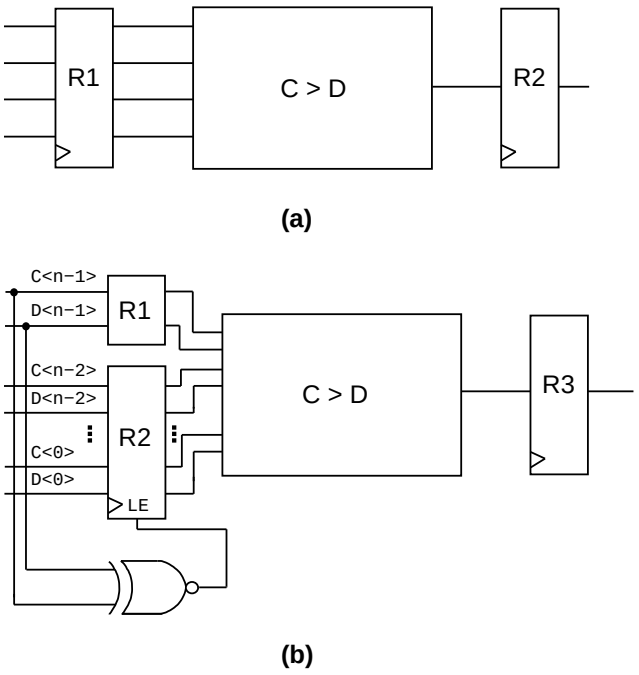

Fig. 1. Precomputation Applied to a Comparator

cycle. For example, when a branch instruction is being executed by a CPU, a multiply unit may not be used. The input registers to the multiplier are maintained at their previous values, ensuring that switching activity power in the multiplier is zero for this clock cycle.

C.4 Precomputation

The gated clock paradigm of the previous section can be carried much further. Given a logic-level circuit and a particular input stimulus, if idling subcircuits can be detected which do not contribute to the computation of the output response for this input stimulus, power reduction can be obtained by "turning off" the idling subcircuits. A technique called precomputation, originally presented in [1], achieves data-dependent power down at the sequential logic or combinational logic level.

In a sequential precomputation architecture, the output logic values of a circuit are selectively precomputed one clock cycle before they are required, and these precomputed values are used to reduce internal switching activity in the succeeding clock cycle. An example of one such architecture applied to a comparator circuit is shown in Figure 1, taken from [1].

The circuit of Figure 1(a) is an n-bit comparator that compares two n-bit numbers C and D and computes the function $C > D$. The circuit with additional precomputation logic is shown in Figure 1(b). The precomputation logic is the logic that is connected to the load enable signal of the registers marked LE.

$$LE = C\langle n-1\rangle \otimes D\langle n-1\rangle$$

where $\otimes$ stands for the exclusive-nor operator.

When the XNOR gate evaluates to a 0, the load enable signal for the registers connected to $C\langle n-2:0\rangle$ and $D\langle n-2:0\rangle$ is turned off. This means that the outputs of these registers do not switch in the next cycle. The correct value for the output is computed even though these input hold possibly erroneous values because either:

- $C\langle n-1\rangle = 1$ and $D\langle n-1\rangle = 0$ in which case the output is a 1 regardless of $C\langle n-2:0\rangle$ and $D\langle n-2:0\rangle$, or
- $C\langle n-1\rangle = 0$ and $D\langle n-1\rangle = 1$ in which case the output is a 0 regardless of $C\langle n-2:0\rangle$ and $D\langle n-2:0\rangle$.

The reduction in power dissipation is a function of the probability that the XNOR gate evaluates to a 0. Other inputs can be added to the precomputation logic to increase power reduction.

If transparent latches are used in the place of flip-flops, the transformation of Figure 1(b) is applicable to combinational circuits. Various sequential and combinational architectures described in [1] have been developed further. Given a combinational circuit, algorithms to determine the subcircuits to be turned off, and the logic required to perform the disabling are presented in [30] and [44]. The techniques of [30] use universal quantification to determine the subcircuits and those of [44] use observability don't-care sets to determine the subcircuits.

A method to reduce switching activity in finite state machines by checking for loop-edges in the State Transition Graph of the machine, and disabling the computation of the next state for these edges is presented in [4].

IV. Architecture or Behavior Level

We survey power analysis methods and optimizations for low power at the architecture level in this section.

A. Architecture Level Power Analysis

The estimation or analysis of the power consumption of a design is a first step towards incorporating power optimization techniques in a synthesis system. Without adequate analysis techniques, it is impossible to evaluate the various designs in the solution space explored during synthesis. Even outside of synthesis, power analysis tools can be of great use to designers, by helping them explore the design space manually. A direct method for power analysis is to translate the given high-level architecture description to the gate, circuit, or physical level; at which point reasonably accurate low-level power analysis tools can be utilized. (For a survey of available tools at the gate level see [31].) This method is obviously infeasible if a large number of design alternatives have to be evaluated, which is the case in synthesis. Reasonable power models, however, can be built if the final lower level circuit style, module and gate library, etc., are fixed, or at the least, restricted in some way. The lower level analysis tools can then be used to create power models for the underlying architecture primitives such as datapath execution units, control units, memory elements, and interconnect. The power models are obtained by characterizing the estimated capacitance that would switch when the given module is activated. This approach is used in [15]. In [21], [22], known signal statistics are used to obtain models that are more accurate than those obtained from using random input streams. What is needed for this is an estimate of the activity for each module. Activity factors for the modules can be obtained from functional simulation over typical input streams, or from statistical/analytical models that are built where possible. An alternate simulation based approach is described in [36] where average power costs are assigned to individual modules, in isolation from other modules. During simulation, the power costs of the modules involved in the given computation are added up. This method ignores the correlations between the activities of different modules.

Other specialized approaches for architecture-level power estimation have been developed. These tend to be less accurate than the above methods, but may be acceptable since they are intended to provide only rough predictions. A model for estimating the power consumption of CMOS chips using gate counts, memory size, logic styles, and layout

styles is described in [41]. A power model to evaluate the power cost of cache options, and multiple function units is developed in [6] and [12], respectively.

Several of the synthesis methods surveyed in the following sections use power models or estimation methods that have been tailored to their application domain and search method.

B. Power Optimizations in Behavioral Synthesis

Behavioral synthesis refers to the process of mapping a high-level specification of a problem into a register-transfer level design. The high-level specification is typically in the form of a data-flow graph and a control-flow graph or a combination of the two. There has been some recent work that has addressed the optimizations for low-power that are possible at this level.

The input high-level specification can be modified through specific transformations that potentially lead to power reduction. The most important transformations for fixed throughput systems are those which reduce the number of control steps. Slower clocks can then be used for the same throughput, enabling the use of lower supply voltages. The quadratic decrease in power consumption can compensate for the additional capacitance introduced due to transformations that increase concurrency. Transformations that reduce the amount of resources needed to implement a given graph can be extended to reduce the amount of capacitance that switches. A number of these transformations are used in an automated system as described in [7]. The transformations are guided by a power estimation method that is based on the parameters of the given data/control flow specification, such as the number of operations of each kind, number of edges, etc. [27]. Specific transformations for DSP circuits are studied in [10].

After the initial specification (data/control flow graph) has been transformed, the individual operations have to be assigned control steps (scheduling) and execution units or modules (allocation and assignment). If a number of modules, with a range of power/delay costs, is available for implementing the given operations types, an appropriate choice of modules can lead to lower power costs for the same performance [17]. The allocation and assignment processes map operations in the control/data flow graph to functional units, variables to registers, and define the interconnect between them in terms of multiplexers and buses. The decisions made during these processes, including the extent of hardware sharing and the sequence of operations (variables) mapped to each functional unit (register), affect the total switched capacitance in the data path. The problem of minimizing this switched capacitance, while accounting for correlations between signals is addressed in [33], [34].

The power consumed in memories can be a major part of the system power consumption. This problem is addressed in [14] in the context of multi-dimensional signal processing subsystems. It is noted that the memories impact power in two ways. First, memory accesses consume a lot of power, especially if the access is off-chip, and second, the greater the size of memory, the greater is the capacitance that switches per access. Control flow transformations, such as loop reordering are presented to try to minimize the memory component of the overall system power consumption.

Several specific design examples illustrate some of the architectural and algorithmic tradeoffs and optimizations that can be used for low power designs. Besides recent issues of the various journals and proceedings that deal with VLSI circuit design, the proceedings of the 1994 IEEE Symposium on Low Power Electronics and the 1994 International Workshop on Low-Power Design are a good source for these design examples.

V. System and Software Level

An increasing fraction of applications are now being implemented as embedded systems. These systems consist of a hardware and a software component. The software component is application-specific software running on a dedicated microprocessor/application specific processor (ASP), while the hardware component consists of application-specific circuits. Hardware-based power estimation and optimization approaches are not completely applicable here, since a major part of the functionality is in the form of instructions as opposed to gates. This motivates the need to consider the power consumption in microprocessors from the point of view of software. This has largely been neglected until recently, since accurate power analysis tools existed only at the circuit or gate level. It is either impractical or impossible to use these to analyze the power consumption over large programs; a preliminary step in alleviating these difficulties is taken in [28] where sequential circuit estimation methods have been extended to handle the case of processors executing specific programs.

These problems can be overcome if the current being drawn by the CPU during the execution of a program is physically measured. An inexpensive and practical technique in this regard has been developed [46] for analyzing the power cost of programs for a given CPU. It has been successfully applied to develop instruction-level power models for some commercial CPUs. While the measurement based technique can only be applied to existing CPUs, its basic methodology can be adapted to use architecture-level power simulators, some of which were described in Section IV-A.

Given the ability to evaluate programs in terms of their power/energy costs, it is possible to search the design space in software power optimization. The choice of the algorithm used can impact the power cost since it determines the runtime complexity of a program. This issue is explored in [49]. Automated tools for synthesizing the optimum algorithm, however, are not available, and this is a very difficult problem. The process of compilation or code-generation can also impact the power cost, since there exist many mappings from a high-level source program into machine code. If power costs of individual instructions are available, an appropriate choice of instructions in the generated code can lead to a reduction in the power cost. This aspect has been studied in the context of specific CPUs [45]. An important lesson learned here is that faster code almost always implies lower energy code. In addition, register allocation can have a significant affect on the power consumed, since register operands are much cheaper than memory operands. This implies that optimizations and transformations aimed at improving performance, and reducing memory accesses will lead to energy efficient code.

It has been noted that the order of instructions can also have an impact on power since it determines the internal switching in the CPU. A scheduling technique has been presented to reduce the estimated switching in the control path of the CPU [40]. Experiments reveal that this may not be an important issue for large general purpose CPUs [46]. However, scheduling of instructions does have an impact in the case of a smaller DSP processor [23]. An additional optimization applicable for this and similar processors is the ability to compact the instruction stream through pairing of instructions.

VI. Summary

We have surveyed power optimizations applicable at various levels of abstraction, namely the circuit, logic, architecture and system level. This survey is not comprehensive, rather we have focused a few typical optimizations at each level of abstraction. Further, device-level and

layout-level optimizations to reduce power have not been presented.

Lowering power dissipation at all abstraction levels is a focus of intense academic and industrial research. These methods are being incorporated into state-of-the-art Computer-Aided Design frameworks.

REFERENCES

[1] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Pa paefthymiou. Precomputation-Based Sequential Logic Optimization for Low Power . *IEEE Transactions on VLSI Systems*, 2(4):426–436, December 1994.

[2] P. Ashar, S. Devadas, and A. R. Newton. *Sequential Logic Synthesis*. Kluwer Academic Publishers, Boston, Massachusetts, 1991.

[3] R. I. Bahar, H. Cho, G. D. Hachtel, E. Macii, and F. Somenzi. A Symbolic Method to Reduce Power Consumption of Circuits Containing False Paths. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 368–371, November 1994.

[4] L. Benini and G. De Micheli. Transformation and Synthesis of FSMs for Low Power Gated Clock Implementation. In *Proceedings of the Int'l Symposium on Low Power Design*, April 1995.

[5] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wan g. MIS: A Multiple-Level Logic Optimization System. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, CAD-6(6):1062–1081, November 1987.

[6] J. Bunda, W. Athas, and D. Fussell. Evaluating power implications of CMOS microprocessor design decisions. In *Proceedings of the International Workshop on Low Power Design*, pages 147–152, Napa, CA, Apr. 1994.

[7] A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. Brodersen. Optimizing power using transformations. *IEEE Transactions on Computer-aided Design*, 14(1), January 1995.

[8] A. Chandrakasan, T. Sheng, and R. W. Brodersen. Low Power CMOS Digital Design. *Journal of Solid State Circuits*, 27(4):473–484, April 1992.

[9] Anantha P. Chandrakasan. *Low-Power Digital CMOS Design*. PhD thesis, University of California at Berkeley, UCB/ERL Memorandum No. M94/65, August 1994.

[10] A. Chatterjee and R. Roy. Synthesis of low power linear DSP circuits using activity metrics. In *International Conference on VLSI Design*, India, Jan. 1994.

[11] W. A. Chren. Low Delay-Power Product CMOS Design Using One-Hot Residue Coding. In *Proceedings of the Int'l Symposium on Low Power Design*, April 1995.

[12] T. Conte, K. Menezes, and S. Sathaye. The impact of power and area efficieny on superscalar processor design. University of South Carolina, Computer Architecture Research Laboratory.

[13] S. Devadas, A. Ghosh, and K. Keutzer. *Logic Synthesis*. McGraw Hill, New York, NY, 1994.

[14] F. Catthoor et al. Global communication and memory optimizing transformations for low power signal processing systems. In *IEEE workshop on VLSI signal processing*, La Jolla, CA, Oct. 1994.

[15] S. Powell et al. Estimating power dissipation of VLSI signal processing chips: The PFA technique. *VLSI Signal Processing IV*, pages 250–259, 1990.

[16] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of Average Switching Activity in Combinational and Sequential Circuits. In *Proceedings of the* 29^{th} *Design Automation Conference*, pages 253–259, June 1992.

[17] L. Goodby, A Orailoglu, and P. Chau. Microarchitectural synthesis of performance-constrained, low-power VLSI designs. In *Proceedings of the International Conference on Computer Design*, pages 323–326, Boston, MA, Oct. 1994.

[18] G. D. Hachtel, M. Hermida, A. Pardo, M. Poncino, and F. Somenzi. Re-Encoding Sequential Circuits to Reduce Power Dissipation. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 70–73, November 1994.

[19] S. Iman and M. Pedram. Multi-Level Network Optimization for Low Power. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 371–377, November 1994.

[20] K. Keutzer. DAGON: Technology Mapping and Local Optimization. In *Proceedings of the* 24^{th} *Design Automation Conference*, pages 341–347, June 1987.

[21] P. Landman and J. Rabaey. Power estimation for high level synthesis. In *Proceedings of the European Design Automation Conference*, pages 361–366, Paris, Feb. 1993.

[22] P. Landman and J. Rabaey. Black-box capacitance models for architectural power analysis. In *Proceedings of the International Workshop on Low Power Design*, pages 165–170, Napa, CA, April 1994.

[23] T. C. Lee, V. Tiwari, S. Malik, and M. Fujita. Power analysis and low-power scheduling techniques for embedded DSP software. Technical Report FLA-CAD-95-01, Fujitsu Labs of America, March 1995.

[24] C. E. Leiserson, F. M. Rose, and J. B. Saxe. Optimizing Synchronous Circuitry by Retiming. In *Proceedings of* 3^{rd} *CalTech Conference on VLSI*, pages 23–36, March 1983.

[25] C. Lemonds and S. S. Mahant Shetti. A Low Power 16 by 16 Multiplier Using Transition Reduction Circuitry. In *Proceedings of the Int'l Workshop on Low Power Design*, pages 139–142, April 1994.

[26] B. Lin. Technology Mapping for Low Power Dissipation. In *Proceedings of the Int'l Conference on Computer Design: VLSI in Computers and Processors*, October 1993.

[27] R. Mehra and J. Rabaey. Behavioral level power estimation and exploration. In *Proceedings of the International Workshop on Low Power Design*, pages 197–204, Napa, CA, April 1994.

[28] J. Monteiro and S. Devadas. Techniques for the Power Estimation of Sequential Logic Circuits Under User-Specified Input Sequences and Programs. In *Proceedings of the Int'l Symposium on Low Power Design*, April 1995.

[29] J. Monteiro, S. Devadas, and A. Ghosh. Retiming Sequential Circuits for Low Power. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 398–402, November 1993.

[30] J. Monteiro, J. Rinderknecht, S. Devadas, and A. Ghosh. Optimization of Combinational and Sequential Logic Circuits for Low Power Using Precomputation. In *Proceedings of the 1995 Chapel Hill Conference on Advanced Research on VLSI*, March 1995.

[31] F. Najm. A Survey of Power Estimation Techniques in VLSI Circuits (*Invited Paper*). *IEEE Transactions on VLSI Systems*, 2(4):446–455, December 1994.

[32] S. C. Prasad and K. Roy. Circuit Optimization for Minimization of Power Consumption Under Delay Constraint. In *Proceedings of the Int'l Workshop on Low Power Design*, pages 15–20, April 1994.

[33] A. Raghunathan and N. Jha. Behavioral synthesis for low power. In *Proceedings of the International Conference on Computer Design*, pages 318–322, Boston, MA, Oct. 1994.

[34] A. Raghunathan and N. Jha. ILP formulation for low power based on minimizing switched capacitance during data path allocation. In *Proceedings of the International Symposium on Circuits & Systems*, 1995.

[35] K. Roy and S. Prasad. SYCLOP: Synthesis of CMOS Logic for Low Power Applications. In *Proceedings of the Int'l Conference on Computer Design: VLSI in Computers and Processors*, pages 464–467, October 1992.

[36] T. Sato, M. Nagamatsu, and H. Tago. Power and performance simulator: ESP and its application for 100MIPS/W class RISC design. In *Proceedings of 1994 IEEE Symposium on Low Power Electronics*, pages 46–47, San Diego, CA, Oct. 1994.

[37] H. Savoj, R. Brayton, and H. Touati. Extracting Local Don't-Cares for Network Optimization. In *Proceedings of the International Conference on Computer-Aided Design*, pages 514–517, November 1991.

[38] A. Shen, S. Devadas, A. Ghosh, and K. Keutzer. On Average Power Dissipation and Random Pattern Testability of Combinational Logic Circuits. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 402–407, November 1992.

[39] M. Stan and W. Burleson. Limited-weight codes for low-power I/O. In *Proceedings of the Int'l Workshop on Low Power Design*, pages 209–214, April 1994.

[40] C. L. Su, C. Y. Tsui, and A. Despain. Saving power in the control path of embedded processors. In *IEEE Design & Test of Computers*, pages 24–30, Winter 1994.

[41] C. Svensson and D. Liu. A power estimation tool and prospects for power savings in CMOS VLSI chips. In *Proceedings of the International Workshop on Low Power Design*, pages 171–176, Napa, CA, Apr. 1994.

[42] C. H. Tan and J. Allen. Minimization of Power in VLSI Circuits Using Transistor Sizing, Input Ordering, and Statistical Power Estimation. In *Proceedings of the Int'l Workshop on Low Power Design*, pages 75–80, April 1994.

[43] V. Tiwari, P. Ashar, and S. Malik. Technology Mapping for Low Power. In *Proceedings of the* 30^{th} *Design Automation Conference*, pages 74–79, June 1993.

[44] V. Tiwari, S. Malik, and P. Ashar. Guarded Evaluation: Pushing Power Management to Logic Synthesis/Design. In *Proceedings of the Int'l Symposium on Low Power Design*, April 1995.

[45] V. Tiwari, S. Malik, and A. Wolfe. "Compilation techniques for low energy: an overview". In *Proceedings of 1994 IEEE Symposium on Low Power Electronics*, pages 38–39, San Diego, CA, Oct. 1994.

[46] V. Tiwari, S. Malik, and A. Wolfe. "Power analysis of embedded software: a first step towards software power minimization". *IEEE Transactions on VLSI Systems*, 2(4):437–445, Dec. 1994.

[47] C-Y. Tsui, M. Pedram, C-A. Chen, and A. M. Despain. Low Power State Assignment Targeting Two- and Multi-level Logic Implementations. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 82–87, November 1994.

[48] C-Y. Tsui, M. Pedram, and A. M. Despain. Technology Decomposition and Mapping Targeting Low Power Dissipation. In *Proceedings of the* 30^{th} *Design Automation Conference*, pages 68–73, June 1993.

[49] P. w. Ong and R. H. Yan. Power-conscious software design - a framework for modeling software on hardware. In *Proceedings of 1994 IEEE Symposium on Low Power Electronics*, pages 36–37, San Diego, CA, Oct. 1994.