High-Level Power Modeling, Estimation, and Optimization

Enrico Macii
Politecnico di Torino
Dip. di Automatica e Informatica
Torino, ITALY 10129

Massoud Pedram
University of Southern California
Dept. of Electrical Eng.–Systems
Los Angeles, CA 90089

Fabio Somenzi
University of Colorado
Dept. of Electrical and Computer Eng.
Boulder, CO 80309

Abstract

In the past, the major concerns of the VLSI designers were area, performance, cost, and reliability. In recent years, however, this has changed and, increasingly, power is being given comparable weight to area and speed. This is mainly due to the remarkable success of personal computing devices and wireless communication systems, which demand high-speed computation and complex functionality with low power consumption. In addition, there exists a strong pressure for manufacturers of high-end products to keep power under control. The main driving factors for lower power dissipation in these products are the costs associated with packaging and cooling, and circuit reliability.

Tools for the automatic design of low-power VLSI systems have thus become mandatory. More specifically, following a natural trend, interests of researchers have lately shifted to the investigation of high-level power modeling, estimation, synthesis, and optimization techniques that account for power dissipation as the primary cost factor.

This paper provides a non-exhaustive survey of the most successful and innovative ideas in this area that have appeared in the literature in the last few years.

1 Introduction

Today's electronic systems are designed starting from specifications given at a very high level of abstraction. This is because many EDA tools accept as input a design expressed in a high-level hardware description language (e.g., VHDL, Verilog), and can automatically produce the corresponding transistor-level implementation with very limited human intervention.

Since power consumption has become a critical issue in the development of digital systems, tools that allow one to control the power budget during the various phases of the design process are in high demand. Given an initial specification of the behavior of the system, several synthesis/optimization steps are required to generate a power-efficient transistor-level description (i.e., a net-list).

DAC 97, Anaheim, California

In order to make the search of the optimal solution as effective as possible, at each level of abstraction an "improvement loop" is used. In such a loop, a power analyzer/estimator ranks the various design, synthesis and optimization options, and thus helps in selecting the one that is potentially more effective from the power standpoint. Obviously, collecting the feed-back on the impact of the different choices on a level-by-level basis, instead of just at the very end of the flow (i.e., at the transistor-level), enables a shorter development time. On the other hand, this paradigm requires the availability of power estimators, as well as synthesis and optimization tools, that provide accurate and reliable results at the various levels of abstraction.

In this paper, we review some of the most relevant contributions to the field of high-level power modeling, estimation, synthesis, and optimization that have appeared in the literature in the last few years.

2 Power Modeling and Estimation

This section describes techniques for power estimation and analysis at different design levels.

2.1 Statistical Sampling

Existing techniques for power estimation at the gate and circuit-level can be divided into two classes: *Static* and *dynamic* [1]. Static techniques [2, 3, 4, 5, 6, 7, 8] rely on probabilistic information about the input stream (such as the mean activity of the input signals and their correlations) to estimate the internal switching activity of the circuit. While these are very efficient, their main limitation is that they cannot accurately capture factors such as slew rates, glitch generation and propagation, and DC fighting. Dynamic techniques [9, 10] explicitly simulate the circuit under a "typical" input stream. They can be applied at both the circuit and gate-level. Their main shortcoming is, however, that they are very slow. Moreover, their results are highly dependent on the simulated sequence. To alleviate this dependence, and thereby produce a trustworthy power estimate, the required number of simulated vectors is usually high, which further exacerbates the run time problem.

To address this problem, a Monte Carlo simulation technique was proposed in [11]. This technique uses an input model based on a Markov process to generate the input stream for simulation. The simulation is performed in an

iterative fashion. In each iteration, a vector sequence of fixed length (called sample) is simulated. The simulation results are monitored to calculate the mean value and variance of the samples. The iteration terminates when some stopping criterion is met. This approach suffers from two major shortcomings. First, the required number of samples, which directly impacts the simulation run time, is approximately proportional to the ratio between the sample variance and the square of the sample mean value. For certain input sequences, this ratio becomes large, thus significantly increasing the simulation run time. Second, there is a general concern about the normality assumption on the sample distribution. Since the stopping criterion is based on such assumption, if the sample distribution significantly deviates from the normal distribution, the simulation may terminate prematurely. Difficult distributions that cause premature termination include bi-modal, multi-modal, and distributions with long or asymmetric tails.

A more efficient sampling procedure, called *Stratified Random Sampling*, is introduced in [12]. The key idea is to partition the population into disjoint subpopulations (called *strata*) in such a way that the power consumption characteristics within each stratum becomes more homogeneous. The units in each sample are then allocated proportional to sizes of the strata. This generally results in a significant reduction in the sampling variance. The stratification itself is based on a low-cost predictor (e.g., zero-delay power estimation) which needs to be evaluated for every unit in the population. Experimental results on a large set of Iscas'85 benchmarks under non-random input sequences show a 10X improvement in the sampling efficiency compared to the technique of [11].

Existing statistical techniques for the estimation of the mean power in sequential circuits consist of two phases: Warm-up period and random sampling. Both of these phases require a large number of simulated vectors. This makes the efficiency of power estimation for sequential circuits much lower than that for combinational circuits [13].

In addition to estimating the mean value of the power dissipation in a circuit, theory of order statistics and stratified sampling techniques have been used to estimate the maximum power dissipation [14] and the cumulative distribution function for the power dissipation [15]. This information is very useful to chip designers who are interested in reliability analysis, and the AC/DC noise analysis.

2.2 Probabilistic Compaction

Another approach for reducing the power simulation time is to compact the given long stream of bit vectors using probabilistic automata [16]. The idea is to build a *Stochastic State Machine* (SSM) which captures the relevant statistical properties of a given, long bit stream, and then excite this machine by a small number of random inputs so that the output sequence of the machine is statistically equivalent to the initial one. The relevant statistical properties denote, for example, the signal and transition probabilities, and first-order spatio-temporal correlations among bits and across consecutive time frames. The procedure then consists of decomposing the SSM into a set of deterministic state machines, and realizing it through SSM synthesis with some auxiliary inputs. The compacted sequence is generated by uniformly random excitement of such inputs.

An improved algorithm for vector compaction is presented in [17]. The foundation of this approach is also probabilistic in nature: It relies on adaptive (dynamic) modeling of binary input streams as first-order Markov sources of information and is applicable to both combinational and sequential circuits. The adaptive modeling technique itself (best known as *dynamic Markov chain modeling*) was recently introduced in the literature on data compression [18] as a candidate to solve various data compression problems. This original formulation is extended in [17] to manage not only correlations among adjacent bits that belong to the same input vector, but also correlations between successive patterns. The model captures completely spatial correlations and first-order temporal correlations and, conceptually, it has no inherent limitation to be further extended to capture temporal dependencies of higher orders.

A hierarchical technique for compacting large sequences of input vectors is presented in [19]. The distinctive feature of this approach is that it introduces hierarchical Markov chain modeling as a flexible framework for capturing not only complex spatio-temporal correlations, but also dynamic changes in the sequence characteristics such as different input modes. The hierarchical Markov model is used to structure the input space into a hierarchy of macro- and micro-states: At the first level in the hierarchy there is a Markov chain of macro-states describing the input modes, whereas at the second level there is a Markov chain of micro-states describing the internal behavior of each input mode.

Results of these approaches show 1-4 orders of magnitude compaction (depending on the initial length and characteristics of the input sequence) with negligible error (i.e. $\leq 5\%$ in most cases) using PowerMill as the simulator. As a peculiar property, note that none of these approaches needs the actual circuit to compact the input sequences.

2.3 RT-Level Power Estimation

Most RT-level power estimation techniques use capacitance models for circuit modules and activity profiles for data or control signals [20, 21, 22]. Such techniques, which are commonly known as *power macro-modeling*, consist of generating circuit capacitance models for some assumed data statistics or properties. The statistics of input data are gathered during behavioral simulation of the circuit.

The *power factor approximation* technique [20] uses an experimentally determined weighting factor to model the average power consumed by a given module over a range of designs. The weakness of this technique is that it does not account for the effect of input data activity on the module power. In contrast, the *stochastic power analysis* technique [21] is based on an activity-sensitive macro-model which maintains that switching activities of high order bits depend on the temporal correlation of data, whereas lower or-

der bits behave randomly. The module is thus completely characterized by its capacitance models in the most significant bit and least significant bit regions. The break-point between the regions is determined based on the applied signal statistics collected from simulation runs. Going further in this direction, one can use a *bitwise data model* as follows:

$$Pwr = 0.5V^2 f \sum_{i}^{n} C_i E_i \quad (1)$$

where n is the number of inputs for the module in question, C_i is the (regression) capacitance for input pin i, and E_i is the switching activity for the i-th pin of the module. The above equation can be made more accurate by including, for example, spatio-temporal correlation coefficients among the circuit inputs; this will however significantly increase the number of variables in the macro-model equation and thus the equation evaluation overhead.

The abovementioned macro-models are (multi-cycle) *cumulative*, in the sense that they can be used to predict the average power under a sequence of input vectors. In some applications, however, it is essential to estimate the circuit power on a cycle-by-cycle basis. Addressing this need, a *cycle-accurate* power macro-model is introduced in [23]. As an example, one can re-write Equation 1 as:

$$Pwr_k = 0.5V^2 f \sum_{i}^{n} C_i E_{i,k} \quad (2)$$

where Pwr_k denotes the power consumption of the module at cycle k, $E_{i,k}$ is the switching activity (it can assume a value of either 0 or 1) for the i-th input of the module at cycle k and is obtained from functional simulation of the system in which the module is placed. The above equation illustrates that macro-modeling can be used to estimate the power consumption at each cycle; this ability is critical to the statistical approach described in [24].

The authors of [23] describe an automatic procedure for cycle-accurate macro-model generation based on statistical sampling for the *training set* design and regression analysis combined with appropriate statistical tests for macro-model variable selection and coefficient calculation. The statistical framework enables prediction of the power value and the confidence level for the predicted power value.

Experimental results show that power macro-models with a relatively small number of input variables (when carefully trained), predict the module power with a typical error of 5-10% for the average power and 10-20% for the cycle power.

2.4 Behavioral-Level Power Estimation

The typical approach for behavioral-level power prediction is to assume some RT-level template and produce estimates based on that assumption. Important choices include hardware partitioning, type of I/O, memory organization, pipeline design, synchronization scheme, bus architecture, and controller complexity. Fortunately, the designers or the environment often provide hints as to what choices should be made.

Three approaches are possible at this level:

1. Do a *quick synthesize* of the circuit and then estimate power using either RT-level or gate-level techniques described previously.

2. Use *profiling* to determine activity factors for various resources. These activities are then plugged into appropriate power macro-model equations.

3. Develop proper analytical models for estimating the switched capacitance as a function of the *circuit activity* and the *circuit complexity*.

The first approach requires the development of a quick synthesis capability which mimics a full synthesis program. This is a difficult problem (especially in the presence of tight timing constraints) which needs further research. In the second approach, static profiling (based on analysis of the behavioral description) or dynamic profiling (based on direct simulation of the behavior under a typical input stream) are used to capture data activity in the circuit [25, 26]. Important statistics include number of operations of a given type, number of busses, register and memory accesses, and number of I/O operations executed within a given period. Instruction-level or behavioral simulators are easily adapted to produce this information while the statistical techniques described previously can be used to improve the efficiency of these simulators. The key problem in the third approach is estimation of the circuit activity and the circuit complexity. Some parameters that relate to circuit activity include the input activity (entropy), the output activity (entropy), and the circuit functionality. Some parameters that influence the circuit complexity include number and type of arithmetic and/or Boolean operations in the behavioral description, number of states and/or transitions in a controller description, and number of cubes (literals) in a minimum sum-of-products (factored-form) expression of a Boolean function. Works reported in [27, 28, 29, 30, 31, 32] are steps in this direction.

3 Synthesis

Power constraints must be taken into account during the various synthesis phases. In this section, we focus on operation scheduling and resource allocation, as well as supply voltage scheduling and control synthesis. We start with a control-data-flow graph (CDFG) description of the design, on which some power-oriented algorithmic transformations have been already applied (see, for example, [33]), and we target the generation of an RT/gate-level implementation whose power dissipation is under control.

3.1 Operation Scheduling

The goal of a scheduling algorithm is to associate each primitive operation appearing in the CDFG with the time interval (also called control step) in which the operation is to be executed so as to satisfy some design constraints.

When the target is power minimization, operations should be scheduled so that resources that are not performing

useful computations in a given control step can be shut down. In other words, the basic objective of a power-oriented scheduling algorithm is to enable, at a higher level of abstraction, power management techniques commonly exploited at the architectural-level (see Section 4.4).

The scheduling algorithm proposed in [34] attempts to assign the operations involved in determining and controlling the flow of the data within the system to the earliest possible time intervals. This allows to establish which computational units are strictly required to determine the final result of each specific computation. Obviously, unused resources can be disabled during the system execution, thus producing a beneficial effect on the global power budget. To reach this goal, mutually exclusive operations are identified in the CDFG and scheduled for execution in time frames occurring after the decision on which unit must be activated has been taken. In this way, all mutually exclusive units, but one, are guaranteed to be shut down during the current computation. In addition, if mutually exclusive operations are scheduled in the same time interval, it may be possible to make them sharing the corresponding resource, thus yielding a potential further power savings.

3.2 Resource Allocation

The objective of resource allocation is to assign registers and functional units to variables and operations in the CDFG, respectively, and to specify the interconnection of the various resources in terms of busses and multiplexors.

Three classes of resources must be considered, namely registers, functional units, and interconnections. Traditionally, the allocation has been carried out separately, one class of resources at a time (serial allocation). Usually, the power consumed by a resource mainly depends on the input switching activity induced by the data being stored or processed. Since, in reality, the patterns flowing in a circuit have specific probability distributions, the way registers and functional units are allocated in the CDFG heavily impacts the switching activities at the interfaces of the resources. Effective, graph-based algorithms for register [35] and functional unit [36] allocation rely on an accurate computation of the probability density functions at the inputs of the various resources, given the probability distributions for the system primary inputs.

Unfortunately, in some cases, serial allocation may result in sub-optimal solutions, i.e., designs using more interconnections than required; it may then be convenient to perform the three operations concurrently (simultaneous allocation). The technique of [37] considers data-dominated designs, and targets a combined minimization of the total circuit capacitance and the switching activities at the inputs of the registers and the functional modules. The first objective is reached by limiting the total number of resources in the final design implementation and by keeping under control the required amount of steering logic and interconnect. The minimization of the input switching activities, on the other hand, is obtained through exploitation of the correlations that may exist between the data words traveling and being stored within the circuit.

Alternatives to the low-power allocation approaches illustrated above are available in the literature [38, 39, 40, 41]; we do not discuss them here for space reasons.

3.3 Multiple Supply Voltage Scheduling

Supplying different voltages to different parts of a chip may reduce the global energy requirements of a design at a very limited cost in terms of algorithmic and/or architectural modifications. This is because the modules of the chip which are part of the critical paths are powered at the maximum allowed voltage, thus preventing any power reduction but, at the same time, avoiding any delay increase; the power consumed by the macro-components that are not on the critical paths, on the other hand, is minimized through proper voltage scaling.

The presence on the same chip of circuitry powered at different voltages imposes the use of level shifters at the boundaries of the various modules. Obviously, the area and power costs due to such shifters must be considered while evaluating the quality of the optimized circuit.

An important phase in the design flow multi-powered systems is that of assigning the most convenient supply voltage, selected from a fixed number of values, to each operation in the CDFG. The problem to be solved is then the scheduling of the supply voltages so as to minimize the power dissipation under throughput/resource constraints.

A solution to the multiple supply voltage scheduling problem has been proposed in [42]. The technique is based on dynamic programming, and it exploits accurate timing and power characterization of the library macro-modules. The method guarantees optimal results for tree CDFGs, but sub-optimal schedules for DAG CDFGs. Additional work on this subject exists; some is reported in [43, 44].

3.4 Control Synthesis

Scheduling and allocation produce a combined description of data-path and control logic. The latter is normally in the form of a transition structure, whose most familiar representation is a FSM or a collection of FSMs. A similar combination of control and data is normally found in the output of synthesis programs that start from RTL descriptions. The translation of the FSMs into a structural description presents opportunities for reducing power consumption and poses corresponding challenges, especially when the control is complex and contains a large number of latches. In this section, we outline an approach that synthesizes a structural description from a state transition graph (STG) suitable as input to logic-level optimization techniques.

For controllers with more than a handful of latches, the explicit representation of the STG is infeasible. Though decomposition of the controller at the behavioral-level may alleviate the problem, optimization opportunities may be lost in the process. For this reason, symbolic techniques based on Binary Decision Diagrams [45] are often applied to the manipulation of large graphs. BDDs are used to represent the transition relation of the graph. This requires a preliminary encoding of the states, which is nor-

mally heuristically derived from the behavioral description. The graph is then subjected to various transformations intended to improve energy efficiency as well as other metrics. (These techniques are reviewed in Section 4.) Finally, a detailed structural description must be produced from the graph.

A direct translation of the transition relation into gates should produce a structure that is relatively close to a good final solution. Otherwise, the succeeding synthesis algorithms are likely to produce sub-optimal results. The problem when the transition relation is represented by a BDD is that the obvious mapping of each BDD node to a multiplexor results in networks that are large, deep, and slow. Among the approaches that overcome this problem, one builds a circuit in which transitions for a given input vector propagate along a single path, which corresponds to the selected path in the BDD; several optimizations are then applied to control the cost of the circuit [46].

Another approach is based on the work of Minato [47]. Zero-suppressed BDDs can represent very large function covers efficiently. Powerful factorization algorithms exist that work on these symbolic covers. It is therefore possible to first flatten the multilevel representation provided by the transition relation BDD and extract from the two-level cover a multilevel network. Factoring can be guided by low-power concerns, but the objective of the symbolic techniques is to provide a link to existing logic-level optimization tools, not to supplant them.

4 Optimization

There are many ways for reducing the power consumption of an architectural description. Techniques based on voltage down-scaling have been in use for a long time. Approaches aiming at reducing the dynamic component of power, on the other hand, are more recent. In this section, we focus on some solutions of the latter class.

4.1 Bus Encoding

It is known that bus capacitances are usually several orders of magnitude higher than those of the internal nodes of a circuit. Consequently, a considerable amount of power can be saved by reducing the number of transitions at the circuit input/output interfaces. This task can be accomplished by encoding the information transmitted over the busses.

The *Bus-Invert code* of [48] is a simple, yet effective, low-power encoding scheme. It works as follows: The Hamming distance between two successive patterns is computed; if it is larger than $N/2$, where N is the bus width, the current address is transmitted with inverted polarity; otherwise, it is transmitted as is. Obviously, a redundant bus line is needed to signal to the receiving end of the bus which polarity is used for the transmission of the incoming pattern. The method guarantees a maximum of $N/2$ transitions per clock cycle, and it performs well when patterns to be transmitted are randomly distributed in time and no information about their correlation is available. For this reason, it is appropriate for data bus encoding.

Another technique, still applicable to data busses, has been proposed in [49]. The basic observation which has originated this work is that using a transition-based encoding instead of a level-based encoding may limit the number of transitions for non-equiprobable input lines. The idea is then to first transform the input patterns in such a way that input lines become as non-equiprobable as possible, and then to apply a transition-based encoding to the so obtained data words before transmission. The transformation of the input patterns is performed by by means of limited-weight (or starvation) codes.

Concerning address busses, other techniques have been explored. They all rely on the well-known fact that the addresses generated by processors in ordinary computing systems are often consecutive; it may then be convenient to adopt the *Gray code* [50, 51] as encoding strategy. This code achieves its asymptotic best performance of a single transition per emitted address when infinite streams of consecutive addresses are considered, and it is optimum only in the class of irredundant codes. If some redundancy is allowed, as for the Bus-Invert approach, better performance can be achieved by resorting to the *T0 code* [52], which requires an extra line to signal when a pair of consecutive addresses is output on the bus. When such line is high, the current bus value is frozen to avoid unnecessary switchings, and the new address is computed directly by the receiver. On the other hand, when two addresses are not consecutive, the redundant line is low, and the bus operates normally. Several variants of the T0 code are possible, some of which may incorporate the Bus-Invert principle to exploit distinctive spectral characteristics of the streams being transmitted.

The motivation for adopting a bus encoding scheme is a reduction of the global power budget; then, the savings achieved through a bus switching activity decrease must not be offset by the power dissipated by the encoding and decoding circuitry which is required at the bus terminals. In addition, bus latency is usually a critical design constraint. Simultaneous optimization of power and timing must then be targeted while synthesizing the logic for address encoding/decoding.

4.2 Control Logic

Given an abstract specification, that is, the STG of the circuit controller, the optimization task consists of modifying and encoding the graph in preparation for logic synthesis. We review these techniques with particular emphasis on those algorithms that can be applied to large circuits. (Those that dissipate non-negligible amounts of energy.) Among the modifications are decomposition and restructuring. Decomposition techniques produce interconnected FSMs from one large FSM, and they fall broadly into two categories: Those based on the algebraic theory of [53], and those based on the identification in the STG of subroutines or coroutines [54]. A subroutine/coroutine corresponds to a fragment of the STG augmented with a wait state. Shutdown techniques (see Section 4.4) can be applied to the the individual machines because only one is active at any point

in time. Both approaches to decomposition try to minimize the activity along the lines connecting the sub-machines, which tend to drive heavier loads. Decomposition naturally helps tackling the complexity issue; however, no decomposition algorithms are currently available that are applicable to STGs with millions of states.

Restructuring of the STG is a generic term that encompasses those graph transformations that preserve equivalence of behavior (or compatibility in the presence of don't care conditions). The best known of such transformations is state minimization. Algorithms are available for the minimization of very large, completely specified FSMs [55]. However, state minimization by itself may have a deleterious effect on both area and energy efficiency, especially for large circuits. It is more advantageous to use the knowledge of the equivalence classes to identify don't care conditions and then use such conditions in conjunction with a cost function that accounts for the desired cost metrics.

The problem of encoding a state transition graph for low-power consumption has received considerable attention. Among the earliest works is [56]. The idea common to this and other encoding methods (see, for example, [57, 58, 59, 60]) is to use the transition probability of a given arc as a (partial) measure of its cost. The problem is thus translated into the embedding of the state transition graph into a hypercube of suitable dimension so that arcs of high cost connect states at low Hamming distance. Standard search techniques can be applied to this combinatorial optimization problem.

When the STG is large, it is normally given in an already encoded form. The problem is then the one of re-encoding. The initial encoding may come from a manual design and therefore it may provide a useful starting point. In general, however, it is not optimal from the power view-point. The main difference between algorithms for re-encoding [61] and those for encoding is in the size of the problems they try to solve (millions of states vs. thousands). To cope with very large graphs, BDD-based techniques are used to manipulate the graphs and sets of states; and the usual algorithms must be reformulated so as to avoid any explicit iteration over states or edges. The computation of the state probabilities can be carried out exactly [62] or by resorting to approximate techniques [63].

So far we have concerned ourselves with the problem of minimizing average power dissipation. The reader interested in peak power reduction is referred to [8].

4.3 Retiming

Changing the position of registers within a circuit so that operations are performed in different clock cycles without changing the overall behavior is known as *retiming* [64]. This technique has been adapted in [65] to reduce power in pipelines. The key point is that a latch positioned at a gate output inhibits the propagation of spurious activity. This observation drives the iterative placement of the latches of the pipeline. The cost function incorporates the number of latches, so as to avoid a too negative impact on area (with the attendant degradation of energy efficiency).

4.4 Shut-Down Techniques

Digital systems usually contain some logic which is not performing useful computations at each clock cycle. Sizable power reductions can then be achieved by shutting down such logic during some proper cycles.

Pre-computation [66, 67] relies on the idea of duplicating part of the logic with the purpose of pre-computing the circuit output values one clock cycle before they are required, and then use these values to reduce the total amount of switching in the circuit during the next clock cycle. In fact, knowing the output values one clock cycle in advance allows the original logic to be turned off during the next time frame, thus eliminating any charging and discharging of the internal capacitances. Obviously, the size of the logic that pre-calculates the output values must be kept under control, since its contribution to the total power balance may offset the savings achieved by blocking the switching inside the original circuit. Several variants to the basic architecture can then be adopted to take care of this problem; in particular, sometimes it may be convenient to resort to partial, rather than global, shut-down, i.e., to select for power management only a (possibly small) subset of the circuit inputs.

An alternative technique, known as *Gated Clocks* [68, 69, 70], provides a way to selectively stop the clock, and thus force the original circuit to make no transition, whenever the computation to be carried out at the next clock cycle is useless. In other words, the clock signal is disabled in accordance to the idle conditions of the logic network. For reactive circuits, the number of clock cycles in which the design is idle in some wait states is usually large. Therefore, avoiding the power waste corresponding to such states may be significant. The logic for the clock management is automatically synthesized from the Boolean function that represents the idle conditions of the circuit. It may well be the case that considering all such conditions result in an excessively large and power consuming circuitry to be added to the original network. It may then be needed to synthesize a simplified function, which dissipates the minimum possible power, and stops the clock with maximum efficiency. The use of gated clocks has the draw-back that the logic implementing the clock gating mechanism is functionally redundant, and this may create major difficulties in testing and verification. Hints on how to design highly-testable gated clock circuits can be found in [71].

Guarded Evaluation [72] is a shut-down strategy that dynamically disables particular computing units any time the results they produce are not used by other system components. Some guard logic, consisting of a transparent latch with an enable signal, is placed at the inputs of the module being power managed. If the unit is active at a given clock cycle, the enable signal lets the unit operate normally. Otherwise, the unit is shut down. The peculiarity of the method is that the signal used to control the guard logic already exists in the original circuit. Therefore, neither additional logic, nor re-synthesis steps are required when resorting to guarded evaluation, as opposed to the cases of pre-computation and gated clocks.

5 Conclusions

Tools for power estimation and optimization at the behavioral and RT-level are younger and, therefore, less developed than those available at the gate and circuit-level. A wealth of research results and a few pioneering commercial tools have appeared nonetheless in the last couple of years. We expect this field to remain quite active in the foreseeable future. New trends and techniques will emerge, some approaches described in this review will consolidate, while others will become obsolete; this in view of technological and strategic changes in the world of consumer microelectronics. In any case, a further increase in the level of abstraction (e.g., system and software) at which power reductions will be targeted should be expected.

References

[1] M. Pedram, "Power Minimization in IC Design: Principles and Applications," *ACM Trans. on Design Automation of Electronic Systems*, Vol. 1, No. 1, pp. 3-56, 1996.

[2] F. Najm, R. Burch, P. Yang, I. Hajj, "Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits," *IEEE Trans. on CAD,* Vol. 9, No. 4, pp. 439-450, 1990.

[3] A. Ghosh, S. Devadas, K. Keutzer, J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits," *ACM/IEEE DAC-29*, pp. 253-259, Anaheim, CA, Jun. 1992.

[4] C-Y. Tsui, M. Pedram, A. M. Despain, "Efficient Estimation of Dynamic Power Dissipation Under a Real Delay Model," *IEEE/ACM ICCAD-93*, pp. 224-228, Santa Clara, CA, Nov. 1993.

[5] F. Najm, "Transition Density: A New Measure of Activity in Digital Circuits," *IEEE Trans. on CAD,* Vol. 12, No. 4, pp. 310-323, 1993.

[6] R. Marculescu, D. Marculescu, M. Pedram, "Efficient Power Estimation for Highly Correlated Input Streams," *ACM/IEEE DAC-32*, pp. 628-634, San Francisco, CA, Jun. 1995.

[7] C-Y. Tsui, J. Monteiro, M. Pedram, S. Devadas, A. M. Despain, B. Lin, "Power Estimation in Sequential Logic Circuits," *IEEE Trans. on VLSI Systems*, Vol. 3, No. 3, pp. 404-416, 1995.

[8] S. Manne, A. Pardo, R. I. Bahar, G. D. Hachtel, F. Somenzi, E. Macii, M. Poncino, "Computing the Maximum Power Cycles of a Sequential Circuit," *ACM/IEEE DAC-32*, pp. 23-28, San Francisco, CA, Jun. 1995.

[9] C. M. Huizer, "Power Dissipation Analysis of CMOS VLSI Circuits by means of Switch-Level Simulation," *IEEE European Solid State Circuits Conf.*, pp. 61-64, 1990.

[10] C. X. Huang, B. Zhang, A-C. Deng, B. Swirski, "The Design and Implementation of PowerMill," *ACM/IEEE ISLPD-95*, pp. 105-110, Dana Point, CA, Apr. 1995.

[11] R. Burch, F. Najm, P. Yang, T. Trick, "A Monte Carlo Approach for Power Estimation," *IEEE Trans. on VLSI Systems*, Vol. 1, No. 1, pp. 63-71, 1993.

[12] C-S. Ding, C-T. Hsieh, Q. Wu, M. Pedram, "Stratified Random Sampling for Power Estimation," *IEEE/ACM ICCAD-96*, San Jose, CA, pp. 577-582, Nov. 1996.

[13] T-L. Chou, K. Roy, "Statistical Estimation of Sequential Circuit Activity," *IEEE/ACM ICCAD-95*, Santa Clara, CA, pp. 34-37, Nov. 1995.

[14] A. Hill, C-C. Teng, S. M. Kang, "Simulation-Based Maximum Power Estimation," *IEEE ISCAS-96*, Vol. IV, pp. 13-16, Atlanta, GA, May 1996.

[15] C-S. Ding, Q. Wu, C-T. Hsieh, M. Pedram, "Statistical Estimation of the Cumulative Distribution Function for Power Dissipation in VLSI Circuits," *ACM/IEEE DAC-34*, Anaheim, CA, Jun. 1997.

[16] D. Marculescu, R. Marculescu, M. Pedram, "Stochastic Sequential Machine Synthesis Targeting Constrained Sequence Generation," *ACM/IEEE DAC-33*, pp. 696-701, Las Vegas, NV, Jun. 1996.

[17] R. Marculescu, D. Marculescu, M. Pedram, "Adaptive Models for Input Data Compaction for Power Simulators," *ACM/IEEE ASPDAC-2*, pp. 391-396, Chiba, Japan, Jan. 1997.

[18] G. V. Cormack, R. N. Horspool, "Data Compression Using Dynamic Markov Modeling," *Computer Journal*, Vol. 30, No. 6, pp. 541-550, 1987.

[19] R. Marculescu, D. Marculescu, M. Pedram, "Power Estimation Using Hierarchical Markov Models," *ACM/IEEE DAC-34*, Anaheim, CA, Jun. 1997.

[20] S. Powell, P. Chau, "Estimating Power Dissipation of VLSI Signal Processing Chips: The PFA Techniques," *IEEE Workshop on VLSI Signal Processing*, Vol. IV, pp. 250-259, 1990.

[21] P. Landman, J. Rabaey, "Power Estimation for High-Level Synthesis," *IEEE EDAC-93*, pp. 361-366, Paris, France, Feb. 1993.

[22] D. Liu, C. Svensson, "Power Consumption Estimation in CMOS VLSI Chips," *IEEE Journal of Solid State Circuits*, Vol. 29, No. 6, pp. 663-670, 1994.

[23] Q. Wu, C-S. Ding, C-T. Hsieh, M. Pedram, "Statistical Design of Macro-Models for RT-Level Power Evaluation," *ACM/IEEE ASPDAC-2*, pp. 523-528, Chiba, Japan, Jan. 1997.

[24] C-T. Hsieh, C-S. Ding, Q. Wu, M. Pedram, "Statistical Sampling and Regression Estimation in Power Macro-Modeling," *IEEE/ACM ICCAD-96*, pp. 583-588, San Jose, CA, Nov. 1996.

[25] A. Chandrakasan, M. Potkonjak, J. Rabaey, R. Brodersen, "Optimizing Power Using Transformations," *IEEE Trans. on CAD,* Vol. 14, No. 1, pp. 12-31, 1995.

[26] N. Kumar, S. Katkoori, L. Rader, R. Vemuri, "Profile-Driven Behavioral Synthesis for Low Power VLSI Systems," *IEEE Design and Test of Computers*, Vol. 12, No. 3, pp. 70-84, 1995.

[27] K. Muller-Glaser, K. Kirsch, K. Neusinger, "Estimating Essential Design Characteristics to Support Project Planning for ASIC Design Management," *IEEE/ACM ICCAD-91*, pp. 148-151, Santa Clara, CA, Nov. 1991.

[28] J. Rabaey, P. Landman, "Activity-Sensitive Architectural Power Analysis for the Control Path," *ACM/IEEE ISLPD-95*, pp. 93-98, Dana Point, CA, Apr. 1995.

[29] E. Macii, M. Poncino, "Exact Computation of the Entropy of a Logic Circuit," *IEEE GLS-VLSI-96*, pp. 123-128, Ames, IA, Mar. 1996.

[30] D. Marculescu, R. Marculescu, M. Pedram, "Information Theoretic Measures for Power Analysis," *IEEE Trans. on CAD*, Vol. 15, No. 6, pp. 599-610, 1996.

[31] M. Nemani, F. Najm, "Towards a High-Level Power Estimation Capability," *IEEE Trans. on CAD,* Vol. 15, No. 6, pp. 588-598, 1996.

[32] A. Lioy, E. Macii, M. Poncino, M. Rossello, "Accurate Entropy Calculation for Large Logic Circuits Based on Output Clustering," *IEEE GLS-VLSI-97*, pp. 70-75, Urbana, IL, Mar. 1997.

[33] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, R. W. Brodersen, "Optimizing Power Using Transformations," *IEEE Trans. on CAD*, Vol. 14, No. 1, pp. 12-31, 1995.

[34] J. Monteiro, S. Devadas, P. Ashar, A. Mauskar, "Scheduling Techniques to Enable Power Management," *ACM/IEEE DAC-33*, pp. 349-352, Las Vegas, NV, Jun. 1996.

[35] J. M. Chang, M. Pedram, "Low Power Register Allocation and Binding," *ACM/IEEE DAC-32*, pp. 29-35, San Francisco, CA, Jun. 1995.

[36] J. M. Chang, M. Pedram, "Module Assignment for Low Power," *IEEE EuroDAC-96*, pp. 376-381, Geneva, Switzerland, Sep. 1996.

[37] A. Raghunathan, N. K. Jha, "Behavioral Synthesis for Low Power," *IEEE ICCD-94*, pp. 318-322, Cambridge, MA, Oct. 1994.

[38] L. Goodby, A. Orailoglu, P. M. Chau, "Microarchitectural Synthesis of Performance-Constrained, Low-Power VLSI Designs," *IEEE ICCD-94*, pp. 323-326, Cambridge, MA, Oct. 1994.

[39] N. Kumar, S. Katkoori, L. Rader, R. Vemuri, "Profile-Driven Behavioral Synthesis for Low-Power VLSI Systems," *IEEE Design and Test of Computers*, Vol. 12, No. 3, pp. 70-84, 1995.

[40] R. San Martin, J. P. Knight, "Optimizing Power in ASIC Behavioral Synthesis," *IEEE Design and Test of Computers*, Vol. 13, No. 2, pp. 58-70, 1996.

[41] R. Mehra, J. Rabaey, "Exploiting Regularity for Low-Power Design," *IEEE/ACM ICCAD-96*, pp. 166-172, San Jose, CA, Nov. 1996.

[42] J. M. Chang, M. Pedram, "Energy Minimization Using Multiple Supply Voltages," *ACM/IEEE ISLPED-96*, pp. 157-162, Monterey, CA, Aug. 1996.

[43] S. Raje, M. Sarrafzadeh, "Variable Voltage Scheduling," *ACM/IEEE ISLPD-95*, pp. 9-14, Dana Point, CA, Apr. 1995.

[44] M. Johnson, K. Roy, "Optimal Selection of Supply Voltages and Level Conversions During Data-Path Scheduling Under Resource Constraints," *IEEE ICCD-96*, pp. 72-77, Austin, TX, Oct. 1996.

[45] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. on Computers*, Vol. 35, No. 8, pp. 677-691, 1986.

[46] L. Lavagno, P. C. McGeer, A. Saldanha, A. L. Sangiovanni-Vincentelli, "Timed Shannon Circuits: A Power-Efficient Design Style and Synthesis Tool," *ACM/IEEE DAC-32*, pp. 254-260, San Francisco, CA, Jun. 1995.

[47] S-I. Minato, "Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems," *ACM/IEEE DAC-30*, pp. 272-277, Dallas, TX, Jun. 1993.

[48] M. R. Stan, W. P. Burleson, "Bus-Invert Coding for Low-Power I/O," *IEEE Trans. on VLSI Systems*, Vol. 3, No. 1, pp. 49-58, 1995.

[49] M. R. Stan, W. P. Burleson, "Limited-Weight Codes for Low-Power," *ACM/IEEE IWLPD-94*, pp. 209-214, Napa Valley, CA, Apr. 1994.

[50] C. L. Su, C-Y. Tsui, A. M. Despain, "Saving Power in the Control Path of Embedded Processors," *IEEE Design and Test of Computers*, Vol. 11, No. 4, pp. 24-30, 1994.

[51] H. Mehta, R. M. Owens, M. J. Irwin, "Some Issues in Gray Code Addressing," *IEEE GLS-VLSI-96*, pp. 178-180, Ames, IA, Mar. 1996.

[52] L. Benini, G. De Micheli, E. Macii, D. Sciuto, C. Silvano, "Asymptotic Zero-Transition Activity Encoding for Address Busses in Low-Power Microprocessor-Based Systems," *IEEE GLS-VLSI-97*, pp. 77-82, Urbana, IL, Mar. 1997.

[53] J. Hartmanis, R. E. Stearns, "Algebraic Structure Theory of Sequential Machines," Prentice-Hall, Englewood Cliffs, NJ, 1966.

[54] S. Devadas, A. R. Newton, "Decomposition and Factorization of Sequential Finite State Machines," *IEEE Trans. on CAD*, Vol. 8, No. 11, pp. 1206-1217, 1989.

[55] B. Lin, A. R. Newton, "Implicit Manipulation of Equivalence Classes Using Binary Decision Diagrams," *IEEE ICCD-91*, pp. 81-85, Cambridge, MA, Oct. 1991.

[56] K. Roy, S. C. Prasad, "Circuit Activity Based Synthesis for Low Power Reliable Operations," *IEEE Trans. on VLSI Systems*, Vol. 1, No. 4, pp. 503-513, 1993.

[57] E. Olson, S. M. Kang, "Low-Power State Assignment for Finite State Machines," *ACM/IEEE IWLPD-94*, pp. 63-68, Napa Valley, CA, Apr. 1994.

[58] C-Y. Tsui, M. Pedram, A. M. Despain, "Low Power State Assignment Targeting Two- and Multi-Level Logic Implementations, *IEEE/ACM ICCAD-94*, San Jose, CA, pp. 82-87, Nov. 1994.

[59] L. Benini, G. De Micheli, "State Assignment for Low Power Dissipation," *IEEE Journal of Solid State Circuits*, Vol. 30, No. 3, pp. 258-268, 1995.

[60] P. Surti, L. F. Chao, A. Tyagi, A. "Low Power FSM Design Using Huffman-Style Encoding," *IEEE EDTC-97*, pp. 521-525, Paris, France, Mar. 1997.

[61] G. D. Hachtel, M. Hermida, A. Pardo, M. Poncino, F. Somenzi, "Re-Encoding Sequential Circuits to Reduce Power Dissipation," *IEEE/ACM ICCAD-94*, pp. 70-73, San Jose, CA, Nov. 1994.

[62] G. D. Hachtel, E. Macii, A. Pardo, F. Somenzi, "Markovian Analysis of Large Finite State Machines," *IEEE Trans. on CAD*, Vol. 15, No. 12, pp. 1479-1493, 1996.

[63] C-Y. Tsui, M. Pedram, A. M. Despain, "Exact and Approximate Methods for Calculating Signal and Transition Probabilities in FSMs," *ACM/IEEE DAC-31*, pp. 18-23, San Diego, CA, Jun. 1994.

[64] C. E. Leiserson, J. B. Saxe, "Retiming Synchronous Circuitry," *Algorithmica*, Vol. 6, No. 1, pp. 5-35, 1991.

[65] J. Monteiro, S. Devadas, A. Ghosh, "Retiming Sequential Circuits for Low Power," *IEEE/ACM ICCAD-93*, pp. 398-402, Santa Clara, CA, Nov. 1993.

[66] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, M. Papaefthymiou, "Precomputation-Based Sequential Logic Optimization for Low Power," *IEEE Trans. on VLSI Systems*, Vol. 2, No. 4, pp. 426-436, 1994.

[67] J. Monteiro, J. Rinderknecht, S. Devadas, A. Ghosh, "Optimization of Combinational and Sequential Circuits for Low Power Using Precomputation," *1995 Chapel Hill Conf. on Advanced Research in VLSI*, pp. 430-444, Chapel Hill, NC, Mar. 1995.

[68] L. Benini, P. Siegel, G. De Micheli, "Automatic Synthesis of Gated Clocks for Power Reduction in Sequential Circuits," *IEEE Design and Test of Computers*, Vol. 11, No. 4, pp. 32-40, 1994.

[69] L. Benini, G. De Micheli, "Transformation and Synthesis of FSMs for Low Power Gated Clock Implementation," *IEEE Trans. on CAD*, Vol. 15, No. 6, pp. 630-643, 1996.

[70] L. Benini, G. De Micheli, E. Macii, M. Poncino, R. Scarsi, "Symbolic Synthesis of Clock-Gating Logic for Power Optimization of Control-Oriented Synchronous Networks," *IEEE EDTC-97*, pp. 514-520, Paris, France, Mar. 1997.

[71] L. Benini, M. Favalli, G. De Micheli, Design for Testability of Gated-Clock FSMs, *IEEE EDTC-96*, pp. 589-596, Paris, France, Mar. 1996.

[72] V. Tiwari, S. Malik, P. Ashar, "Guarded Evaluation: Pushing Power Management to Logic Synthesis/Design," *ACM/IEEE ISLPD-95* pp. 221-226, Dana Point, CA, Apr. 1995.