Power Simulation and Estimation in VLSI Circuits

Massoud Pedram
University of Southern California
Department of EE-Systems
Los Angeles, CA 90089
pedram@ceng.usc.edu

Abstract

In the past, the major concerns of the VLSI designer were area, speed, and cost; power consideration was typically of secondary importance. In recent years, however, this has begun to change and, increasingly, power is being given comparable weight to other design considerations. Several factors have contributed to this trend, including the remarkable success and growth of the class of battery-powered, personal computing devices and wireless communications systems that demand high-speed computation and complex functionality with low power consumption. In these applications, extending the battery service life is a critical design concern. There also exists a significant pressure for producers of high-end products to reduce their power consumption. The main driving factors for lower power dissipation in these products are the cost associated with packaging and cooling as well as the circuit reliability.

System designers have started to respond to the requirement of power-constrained system designs by a combination of architectural improvements and advanced design automation methodologies and technqiues for low power. In parallel, researchers and CAD tool developers have introduced a variety of models, algorithms, and techniques for estimating the power dissipation in VLSI circuits and systems in support of the low power optimization and synthesis techniques.

This article describes representative contributions to the power modeling and estimation of VLSI circuits at various levels of design abstraction.

1 Introduction

In the past, the major concerns of the VLSI designer were area, performance, cost and reliability; power considerations were mostly of only secondary importance. In recent years, however, this has begun to change and, increasingly, power is being given comparable weight to area and speed. Several factors have contributed to this trend. Perhaps the primary driving factor has been the remarkable success and growth of the class of personal computing devices (portable desktops, audio- and video-based multimedia products) and wireless communications systems (personal digital assistants and personal communicators) which demand high-speed computation and complex functionality with

low power consumption. There also exists a strong pressure for producers of high-end products to reduce their power consumption.

When the target is a low-power application, the search for the optimal solution must include, at each level of abstraction, a "design improvement loop". 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 stand-point. 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 gate-level), enables a shorter development time. On the other hand, this paradigm requires the availability of power simulators and estimators, as well as synthesis and optimization tools, that provide accurate and reliable results at various levels of abstraction.

It has been pointed out in the introduction that the availability of level-by-level power analysis and estimation tools that are able to provide fast and accurate results are key for increasing the effectiveness of automatic design frameworks organized. We start this section, with a concise description of techniques for software-level estimation (Section 2). We then move to the behavioral level (Section 3), where we discuss existing power estimation approaches that rely on information-theoretic (Section 3.1), complexity-based (Section 3.2), and synthesis-based (Section 3.3) models. Finally, we focus our attention to designs described at the RT-level (Section 4). This is the area where most of the research activity on power modeling and estimation has been concentrated in recent times; we cover two of the most investigated classes of methods, namely, those relying on regression-based models (Section 4.1) and on sampling-based models (Section 4.2). Finally, we move to the gate-level power estimation (Section 5), where we discuss existing dynamic power estimation approaches that rely on statistical sampling (Section 5.1) and probabilistic compaction (Section 5.2) as well as probability-based signal propagation schemes (Section 5.3).

This review article may be supplemented with other surveys on the topic, including [1, 2, 3, 4].

2 Software-Level Power Estimation

The first task in the estimation of power consumption of a digital system is to identify the typical application programs that will be executed on the system. A non-trivial application program consumes millions of machine cycles, making it nearly impossible to perform power estimation using the complete program at, say, the RT-level. Most of the reported results are based on *power macro-modeling*, an estimation approach which is extensively used for behavioral and RT-level estimation (see Sections 3 and 4).

In [5], the power cost of a CPU module is characterized by estimating the average capacitance that would switch when the given CPU module is activated. In [6], the switching activities on (address, instruction, and data) buses are used to estimate the power consumption of the microprocessor. In [7], based on actual current measurements

of some processors, Tiwari *et al.* present the following instruction-level power model:

$$Energy_p = \sum_i (BC_i N_i) + \sum_{i,j} (SC_{i,j} N_{i,j}) + \sum_k OC_k$$

where $Energy_p$ is the total energy dissipation of the program which is divided into three parts. The first part is the summation of the base energy cost of each instruction (BC_i is the base energy cost and N_i is the number of times instruction i is executed). The second part accounts for the circuit state ($SC_{i,j}$ is the energy cost when instruction i is followed by j during the program execution). Finally, the third part accounts for energy contribution OC_k of other instruction effects such as stalls and cache misses during the program execution.

In [8], Hsieh *et al.* present a new approach, called *profile-driven program synthesis*, to perform RT-level power estimation for high performance CPUs. Instead of using a macro-modeling equation to model the energy dissipation of a microprocessor, the authors use a synthesized program to exercise the microprocessor in such a way that the resulting instruction trace behaves (in terms of performance and power dissipation) much the same as the original trace. The new instruction trace is however much shorter than the original one, and can hence be simulated on a RT-level description of the target microprocessor to provide the power dissipation results quickly.

Specifically, this approach consists of the following steps:

1. Perform architectural simulation of the target microprocessor under the instruction trace of typical application programs.

2. Extract a *characteristic profile*, including parameters such as the instruction mix, instruction/data cache miss rates, branch prediction miss rate, pipeline stalls, etc. for the microprocessor.

3. Use mixed integer linear programming and heuristic rules to gradually transform a generic program template into a fully functional program.

4. Perform RT-level simulation of the target microprocessor under the instruction trace of the new synthesized program .

Notice that the performance of the architectural simulator in gate-vectors/second is roughly 3 to 4 orders of magnitude higher than that of a RT-level simulator.

This approach has been applied to the Intel Pentium processor (which is a super-scalar pipelined CPU with 8KB 2-way set-associative data, instruction and data caches, branch prediction and dual instruction pipeline) demonstrating 3 to 5 orders of magnitude reduction in the RT-level simulation time with negligible estimation error.

3 Behavioral-Level Power Estimation

Conversely from some of the RT-level methods that will be discussed in Section 4, estimation techniques at the behavioral-level cannot rely on information about the gate-

level structure of the design components, and hence, must resort to abstract notions of physical capacitance and switching activity to predict power dissipation in the design.

3.1 Information-Theoretic Models

Information theoretic approaches for high-level power estimation [9, 10] depend on information theoretic measures of activity (for example, entropy) to obtain quick power estimates.

Entropy characterizes the randomness or uncertainty of a sequence of applied vectors and thus is intuitively related to switching activity, that is, if the signal switching is high, it is likely that the bit sequence is random, resulting in high entropy. Suppose the sequence contains t distinct vectors and let p_i denote the occurrence probability of any vector v in the sequence. Obviously, $\sum_{i=1}^{t} p_i = 1$. The entropy of the sequence is given by:

$$h = -\sum_{i=1}^{t} p_i \log p_i$$

where $\log x$ denotes the base 2 logarithm of x. The entropy achieves its maximum value of $\log t$ when $p_i = 1/t$. For an n-bit vector, $t \leq 2^n$. This makes the computation of the exact entropy very expensive. Assuming that the individual bits in the vector are independent, then we can write:

$$h = -\sum_{i=1}^{n} (q_i \log q_i + (1 - q_i) \log(1 - q_i))$$

where q_i denotes the signal probability of bit i in the vector sequence. Note that this equation is only an upperbound on the exact entropy, since the bits may be dependent. This upperbound expression is, however, the one that is used for power estimation purposes. Furthermore, in [9] it has been shown that, under the temporal independence assumption, the average switching activity of a bit is upper-bounded by one half of its entropy.

The power dissipation in the circuit can be approximated as:

$$Power = 0.5V^2 f C_{tot} E_{avg}$$

where C_{tot} is the total capacitance of the logic module (including gate and interconnect capacitances) and E_{avg} is the average activity of each line in the circuit which is, in turn, approximated by one half of its average entropy, h_{avg}. The average line entropy is computed by abstracting information obtained from a gate-level implementation. In [10], it is assumed that the word-level entropy per logic level reduces quadratically from circuit inputs to circuit outputs, whereas in [9] it is assumed that the bit-level entropy from one logic level to next decreases in an exponential manner. Based on these assumptions, two different computational models are obtained.

In [9], Marculescu *et al.* derive a closed-form expression for the average line entropy for the case of a linear gate distribution, i.e., when the number of nodes scales linearly

between the number of circuit inputs, n, and circuit outputs, m. The expression for h_{avg} is given by:

$$h_{avg} = \frac{2nh_{in}}{(n+m)\ln\frac{h_{in}}{h_{out}}}\left(1 - \frac{m}{n}\frac{h_{out}}{h_{in}} - \frac{(1-\frac{m}{n})(1-\frac{h_{out}}{h_{in}})}{\ln\frac{h_{in}}{h_{out}}}\right)$$

where h_{in} and h_{out} denote the average bit-level entropies of circuit inputs and outputs, respectively. h_{in} is extracted from the given input sequence, whereas h_{out} is calculated from a quick functional simulation of the circuit under the given input sequence or by empirical entropy propagation techniques for pre-characterized library modules. In [10], Nemani and Najm propose the following expression for h_{avg}:

$$h_{avg} = \frac{2}{3(n+m)}(H_{in} + H_{out})$$

where H_{in} and H_{out} denote the average sectional (word-level) entropies of circuit inputs and outputs, respectively. The sectional entropy measures H_{in} and H_{out} may be obtained by monitoring the input and output signal values during a high-level simulation of the circuit. In practice, however, they are approximated as the summation of individual bit-level entropies, h_{in} and h_{out}.

If the circuit structure is given, the total module capacitance is calculated by traversing the circuit netlist and summing up the gate loadings. Wire capacitances are estimated using statistical wire load models. Otherwise, C_{tot} is estimated by quick mapping (for example, mapping to 3-input universal gates) or by information theoretic models that relate the gate complexity of a design to the difference of its input and output entropies. One such model proposed by Cheng and Agrawal in [11], for example, estimates C_{tot} as:

$$C_{tot} = \frac{m}{n}2^n h_{out}$$

This estimate tends to be too pessimistic when n is large; hence, in [12], Ferrandi *et al.* present a new total capacitance estimate based on the number N of nodes (i.e., 2-to-1 multiplexors) in the *Ordered Binary Decision Diagrams* (OBDD) [13] representation of the logic circuit as follows:

$$C_{tot} = \alpha\frac{m}{n}N h_{out} + \beta$$

The coefficients of the model are obtained empirically by doing linear regression analysis on the total capacitance values for a large number of synthesized circuits.

Entropic models for the controller circuitry are proposed by Tyagi in [14], where three entropic lower bounds on the average Hamming distance (bit changes) with state set S and with T states, are provided. The tightest lower bound derived in this paper for a sparse finite state machine (FSM) (i.e., $t \leq 2.23T^{1.72}/\sqrt{\log T}$, where t is the total number of transitions with nonzero steady-state probability) is the following:

$$\sum_{s_i,s_j \in S} p_{i,j}H(s_i,s_j) \geq h(p_{i,j}) - 1.52\log T - 2.16 + 0.5\log(\log T)$$

where $p_{i,j}$ is the steady-state transition probability from s_i to s_j, $H(s_i, s_j)$ is the Hamming distance between the two states, and $h(p_{i,j})$ is the entropy of the probability distribution $p_{i,j}$. Notice that the lower bound is valid regardless of the state encoding used.

In [15], using a Markov chain model for the behavior of the states of the FSM, the authors derive theoretical lower and upper bounds for the average Hamming distance on the state lines which are valid irrespective of the state encoding used in the final implementation. Experimental results obtained for the mcnc'91 benchmark suite show that these bounds are tighter than the bounds reported [14].

3.2 Complexity-Based Models

These models relate the circuit power dissipation to some notion of *circuit complexity*. Example parameters that influence the circuit complexity include the number and the type of arithmetic and/or Boolean operations in the behavioral description, the number of states and/or transitions in a controller description, and the number of cubes (literals) in a minimum sum-of-products (factored-form) expression of a Boolean function.

Most of the proposed complexity-based models rely on the assumption that the complexity of a circuit can be estimated by the number of "equivalent gates". This information may be generated on-the-fly using analytical predictor functions, or retrieved from a pre-characterized high-level design library. An example of this technique is the *chip estimation system* [16], which uses the following expression for the average power dissipation of a logic module:

$$Power = fN(Energy_{gate} + 0.5V^2 C_{load})E_{gate}$$

where f is the clock frequency, N is the gate equivalent count for the component, $Energy_{gate}$ is the average internal consumption for an equivalent gate (it includes parasitic capacitance contributions as well as short-circuit currents) per logic transition, C_{load} is the average capacitive load for an equivalent gate (it includes fanout load capacitances and interconnect capacitances), and E_{gate} is the average output activity for an equivalent gate per cycle. C_{load} is estimated statistically based on the average fanout count in the circuit and custom wire load models. E_{gate} is dependent on the functionality of the module. The data is pre-calculated and stored in the library and is independent of the implementation style (static vs. dynamic logic, clocking strategy), library-specific parameters (gate inertia, glitch generation and propagation), and the circuit context in which the module is instantiated. This is an example of an implementation-independent and data-independent power estimation model.

In [17], Nemani and Najm present a high-level estimation model for predicting the area of an optimized single-output Boolean function. The model is based on the assumption that the area complexity of a Boolean function f is related to the distribution of the sizes of the on-set and off-set of the function. For example, using the "linear

measure", the area complexity of the on-set of f is written as:

$$\mathcal{C}_1(f) = \sum_{i=1}^{N} c_i p_i$$

where the set of integers $\{c_1, c_2, \cdots, c_N\}$ consists of the distinct sizes of the essential prime implicants of the on-set and weight p_i is the probability of the set of all minterms in the on-set of f which are covered by essential primes of size c_i, but not by essential primes of any larger size. The area complexity of the off-set of f $\mathcal{C}_0(f)$ is similarly calculated. Hence, the area complexity of function f is estimated as:

$$\mathcal{C}(f) = \frac{\mathcal{C}_1(f) + \mathcal{C}_0(f)}{2}.$$

The authors next derive a family of regression curves (which happen to have exponential form) relating the actual area $\mathcal{A}(f)$ of random logic functions optimized by the SIS program (in terms of the number of gates) to the area complexity measure $\mathcal{C}(f)$ for different output probabilities of function f. These regression equations are subsequently used for total capacitance estimation and hence high-level power estimation. The work is extended in [18] to area estimation of multiple-output Boolean functions.

A similar technique would rely on predicting the quality of results produced by EDA flows and tools. The predictor function is obtained by performing regression analysis on a large number of circuits synthesized by the tools and relating circuit-specific parameters and/or design constraints to post-synthesis power dissipation results. For example, one may be able to produce the power estimate for an unoptimized Boolean network by extracting certain structural properties of the underlying directed acyclic graph, average complexity of each node, and user-specified constraints and plugging these values in the predictor function.

Complexity-based power prediction models for controller circuitry have been proposed by Landman and Rabaey in [19]. These techniques provide quick estimation of the power dissipation in a control circuit based on the knowledge of its target implementation style (that is, pre-charged pseudo-NMOS or dynamic PLA), the number of inputs, outputs, states, and so on. The estimates will have a higher degree of accuracy by introducing empirical parameters that are determined by curve fitting and least squared fit error analysis on real data. For example, the power model for an FSM implemented in standard cells is given by:

$$Power = 0.5V^2 f(N_I C_I E_I + N_O C_O E_O) N_M$$

where N_I and N_O denote the number of external input plus state lines and external output plus state lines for the FSM, C_I and C_O are regression coefficients which are empirically derived from low-level simulation of previously designed standard cell controllers, E_I and E_O denote the switching activities on the external input plus state lines and external output plus state lines, and finally N_M denotes the number of minterms in an optimized cover of the FSM. Dependence on N_M indicates that this model requires a partial (perhaps symbolic) implementation of the FSM.

3.3 Synthesis-Based Models

One approach for behavioral-level power prediction is to assume some RT-level template and produce estimates based on that assumption. This approach requires the development of a *quick synthesis* capability which makes some behavioral choices (mimicking a full synthesis program). Important behavioral choices include type of I/O, memory organization, pipelining issues, synchronization scheme, bus architecture, and controller design. This is a difficult problem, especially in the presence of tight timing constraints. Fortunately, designers or the environment often provide hints on what choices should be made. After the RT-level structure is obtained, the power is estimated by using any of the RT-level techniques that will be described in Section 4.

Relevant data statistics such as the number of operations of a given type, bus and memory accesses, and I/O operations, are captured by *static profiling* based on stochastic analysis of the behavioral description and data streams [20, 21], or *dynamic profiling* based on direct simulation of the behavior under a typical input stream [22, 23]. Instruction-level or behavioral simulators are easily adapted to produce this information.

4 RT-Level Power Estimation

Most RT-level power estimation techniques use regression-based, switched capacitance models for circuit modules. Such techniques, which are commonly known as *power macro-modeling*, are reviewed next.

4.1 Regression-Based Models

A typical RT-level power estimation flow consists of the following steps:

1. Characterize every component in the high-level design library by simulating it under pseudo-random data and fitting a multi-variable regression curve (i.e., the power macro-model equation) to the power dissipation results using a least mean square error fit [24].

2. Extract the variable values for the macro-model equation from either static analysis of the circuit structure and functionality, or by performing a behavioral simulation of the circuit. In the latter case, a power co-simulator linked with a standard RT-level simulator can be used to collect input data statistics for various RT-level modules in the design.

3. Evaluate the power macro-model equations for high-level design components which are found in the library by plugging the parameter values in the corresponding macro-model equations.

4. Estimate the power dissipation for random logic or interface circuitry by simulating the gate-level description of these components, or by performing probabilistic power estimation. The low level simulation can be significantly sped up by the

application of statistical sampling techniques or automata-based compaction techniques.

The macro-model for the components may be parameterized in terms of the input bit width, the internal organization/architecture of the component, and the supply voltage level. Notice that there are cases where the construction of the macro-model of step (1) can be done analytically using the information about the structure of the gate-level description of the modules, without resorting to simulation as proposed by Benini *et al.* in [25]. On the other hand, if the low-level netlist of the library components is not known (which may be the case for soft macros), step (1) can be replaced by data collection from past designs of the component followed by appropriate process technology scaling [26]. In addition, the macro-model equation in step (2) may be replaced by a table-look up with necessary interpolation equations.

In the following paragraphs, we review various power macro-model equations which exhibit different levels of accuracy versus computation/information usage tradeoff.

The simplest power macro-model, known as the *power factor approximation* technique [27], is a *constant type model* which uses an experimentally determined weighting factor to model the average power consumed by a given module per input change. For example, the power dissipation of an $n \times n$ bit integer multiplier can be written as:

$$Power = 0.5V^2 n^2 C f_{activ}$$

where V is the supply voltage level, C is the capacitive regression coefficient, and f_{activ} is the activation frequency of the module (this should not be confused with the average, bit-level switching activity of multiplier inputs).

The weakness of this technique is that it does not account for the data dependency of the power dissipation. For example, if one of the inputs to the multiplier is always 1, we would expect the power dissipation to be less than when both inputs are changing randomly. In contrast, the *stochastic power analysis* technique proposed by Landman and Rabaey in [28] is based on an activity-sensitive macro-model, called the *dual bit type model*, which maintains that switching activities of high order bits depend on the temporal correlation of data, whereas lower order bits behave randomly. The module is thus completely characterized by its capacitance models in the sign and white noise bit regions. The macro-model equation form is then given by:

$$Power = 0.5V^2 f(n_u C_u E_u + n_s \sum_{xy=++}^{--} C_{xy} E_{xy})$$

where C_u and E_u represent the capacitance coefficient and the mean activity of the unsigned bits of the input sequence, while C_{xy} and E_{xy} denote the capacitance coefficient and the transition probability for the sign change xy in the input stream. n_u and n_s represent the number of unsigned and sign bits in the input patterns, respectively. Note that E_u, E_{xy}, and the boundary between sign and noise bits are determined based on

the applied signal statistics collected from simulation runs. Expanding this direction, one can use a *bitwise data model* as follows:

$$Power = 0.5V^2 f \sum_{i=1}^{n} C_i E_i$$

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. This equation can produce more accurate results 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.

Accuracy may be improved (especially, for components with deep logic nesting, such as multipliers) by power macro-modeling with respect to both the average input and output activities (the *input-output data model*), that is:

$$Power = 0.5V^2 f(C_I E_I + C_O E_O)$$

where C_I and C_O represent the capacitance coefficients for the mean activities of the input and output bits, respectively. The dual bit type model or the bitwise data model may be combined with the input-output data model to create a more accurate, but more expensive, macro-model form. Recently, in [29], the authors presented a 3D-table, power macro-modeling technique which captures the dependence of power dissipation in a combinational logic circuit on the average input signal probability, the average switching activity of the input lines, and the average (zero-delay) switching activity of the output lines. The latter parameter is obtained from a fast functional simulation of the circuit. The paper also presents an automatic macro-model construction procedure based on random sampling principles. Note that the equation form and variables used for every module are the same (although the regression coefficients are different).

A parametric power model is described by Liu and Svensson in [30], where the power dissipation of the various components of a typical processor architecture, including on-chip memory, busses, local and global interconnect lines, H-tree clock net, off-chip drivers, random logic, and data path, are expressed as a function of a set of parameters related to the implementation style and internal architecture of these components. For example, consider a typical on-chip memory (a storage array of 6-transistor memory cells) which consists of four parts: The memory cells, the row decoder, the column selection, the read/write circuits. The power model for a cell array of 2^{n-k} rows and 2^k columns in turn consists of expressions for: (1) the power consumed by 2^k memory cells on a row during one pre-charge or one evaluation; (2) the power consumed by the row decoder; (3) the power needed for driving the selected row; (4) the power consumed by the column select part; and (5) the power dissipated in the sense amplifier and the readout inverter. For instance, the memory cell power (expression 1 in above) is given by:

$$Power_{memcell} = 0.5VV_{swing}2^k(C_{int} + 2^{n-k}C_{tr})$$

where V_{swing} is the voltage swing on the bit/$\overline{bit}$ line (which may be different for read versus write), C_{int} gives the wiring-related row capacitance per memory cell, and $2^{n-k}C_{tr}$ gives the total drain capacitances on the bit/$\overline{bit}$ line. Notice that during the read time, every memory cell on the selected row drives exactly bit or $\overline{bit}$.

A salient feature of the above macro-model techniques is that they only provide information about average power consumption over a relatively large number of clock cycles. The above techniques, which are suitable for estimating the average-power dissipation, are referred to as *cumulative* power macro-models. In some applications, however, estimation of average power only is not sufficient. Examples are circuit reliability analysis (maximum current limits, heat dissipation and temperature gradient calculation, latch-up conditions), noise analysis (resistive voltage drop and inductive bounce on power and ground lines), and design optimization (power/ground net topology design, number and placement of decoupling capacitors, buffer insertion, etc.). In these cases, *cycle-accurate* (*pattern-accurate*) power estimates are required.

Mehta *et al.* propose a clustering approach for pattern-accurate power estimation in [31]. This approach relies on the assumption that closely related input transitions have similar power dissipation. Hence, each input pattern is first mapped into a cluster, and then a table look-up is performed to obtain the corresponding power estimates from pre-calculated and stored power characterization data for the cluster. The weakness of this approach is that, for efficiency reasons, the number of clusters has to be relatively small, which would introduce errors into the estimation result. In addition, the assumption that closely related patterns (e.g., patterns with short Hamming distance) result in similar power distribution may be quite inaccurate, especially when the *mode-changing bits* are involved, i.e., when a bit change may cause a dramatic change in the module behavior.

Addressing these problems, Wu *et al.* describe in [32] 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 (i.e., the F^* test) for macro-model variable selection and coefficient calculation. The test identifies the most (least) power-critical variable to add to (delete from) the set of selected variables. The statistical framework enables prediction of the power value and the confidence level for the predicted power value. This work is extended by Qiu *et al.* in [33] to capture "important" first-order temporal correlations and spatial correlations of up to order three at the circuit inputs. Note that here the equation form and variables used for each module are unique to that module type. Experimental results show that power macro-models with a relatively small number of input variables (i.e., 8) predict the module power with a typical error of 5-10% for the average power and 10-20% for the cycle power.

The abovementioned high-level power macromodeling techniques are mostly applicable to the combinational modules (such as adders, multipliers) in the circuit. This is because the power consumption in a combinational circuit module can be fully determined from the statistical characteristics of the vector sequence applied to its primary inputs. As mentioned above, in some cases, it is beneficial to use variables related to

the vectors which appear on the primary outputs to further improve the accuracy of the power macromodel for combinational modules. For sequential circuit modules, using only variables related to the external inputs and outputs is not enough to model the power consumption accurately; This is because power consumption in sequential circuits strongly depends on the state transitions, which are not visible from outside. To generate an accurate power model for the sequential module, one thus needs to include some variables related to the internal state of the module.

For complex intellectual property (IP) cores, generating a power macro-model is even more challenging because of the following:

- The IP core may have internal state variables; it is therefore very difficult to generate an accurate power model for the IP core if we have access to information about its primary inputs and primary outputs only. Therefore, it is desirable to construct a power macromodel for the IP core that includes variables related to not only its primary inputs/outputs, but also its internal state variables.

- At the system level, IP cores are usually specified behaviorally through input/output behavioral modeling. Power modeling however requires information about the internal states of the IP core. Therefore, it is necessary to modify the IP core specification at the system level so that the power model and power evaluation tool can retrieve the information they need for power estimation and optimization.

- There may be thousands of internal state variables in the IP core. Using all of these variables will result in a power model which is too large to store in the system library and too expensive to evaluate. Furthermore, requiring information about all internal states greatly increases the complexity of IP core specification and puts undue burden on the IP vendor to reveal details about the IP core, which can otherwise be hidden from the IP user.

To solve these problems, one needs to develop an IP power model constructor which will automatically generate power macromodels for IP cores using the minimum number of internal state variables. This may be achieved by using a statistical relevance testing procedure such that only the "important" variables are retained in the power model. In this way, a power macromodel with small number of variables, yet sufficient accuracy, can be constructed.

4.2 Sampling-Based Models

RT-level power evaluation can be implemented in the form of a *power co-simulator* for standard RT-level simulators. The co-simulator is responsible for collecting input statistics from the output of the behavioral simulator and producing the power value at the end. If the co-simulator is invoked by the RT-level simulator every simulation cycle to collect activity information in the circuit, it is called *census macro-modeling*.

Evaluating the macro-model equation at each cycle during the simulation is actually a census survey. The overhead of data collection and macro-model evaluation can be

high. To reduce the run time overhead, Hsieh *et al.* use *simple random sampling* to select a sample and calculate the macro-model equation for the vector pairs in the sample only [34]. The sample size is determined before simulation. The *sampler macro-modeling* randomly selects n cycles and marks those cycles. When the behavioral simulator reaches the marked cycle, the macro-modeling invokes the behavioral simulator for the current input vectors and previous input vectors for each module. The input statistics is only collected in these marked cycles. Instead of selecting only one sample of large size, we can select several samples of at least 30 units (to insure normality of sample distribution) before the simulation. Then the average value of sample means is the estimate of population mean. In this manner, the overhead of collecting input statistics at every cycle which is required by census macro-modeling is substantially reduced. Experimental results show that sampler macro-modeling results in an average efficiency improvement of 50X over the census macro-modeling with and average error of 1%.

The macro-model equation is developed by using a training set of input vectors. The training set satisfies certain assumptions such as being pseudo-random data, speech data, etc. Hence, the macro-model may become biased, meaning that it produces very good results for the class of data which behave similarly to the training set; otherwise, it produces poor results. One way to reduce the gap between the power macro-model equation and the gate-level power estimation is to use a *regression estimator* as follows [34]. It can be shown that the plot of the gate-level power value versus a well-designed macro-model equation estimate for many functional units reveals an approximately linear relationship. Hence, the macro-model equation can be used as a predictor for the gate-level power value. In other words, the sample variance of the ratio of gate-level power to macro-model equation power tends to be much smaller than that of the gate-level power by itself. It is thus more efficient to estimate the mean value of this ratio and then use a linear regression equation to calculate the mean value of the circuit-level power. The *adaptive macro-modeling* thus invokes a gate-level simulator on a small number of cycles to improve the macro-model equation estimation accuracy. In this manner, the "bias" of the static macro-models is reduced or even eliminated. Experimental results show that the census macro-modeling incurs large error (an average of 30% for the benchmark circuits) compared to gate level simulation. The adaptive macro-modeling however exhibits an average error of only 5% which demonstrates the superiority of the adaptive macro-modeling technique.

5 Gate-Level Power Estimation

In CMOS circuits, power is mostly consumed during logic switching. This simplifies the power estimation problem to that of calculating the toggle count of each circuit node under a gate level delay model. Several gate level power estimation techniques have been proposed in the past. These techniques which offer orders of magnitude speed-up compared to the conventional simulation-based techniques at the circuit level, can be classified into two classes: *dynamic* and *static*. Dynamic techniques explicitly simulate

the circuit under a "typical" input vector sequence (or input stream). The potential problem with these approaches is that the estimation results strongly depend on the input vector sequence, a phenomenon often described as *pattern dependence*. The static techniques do not explicitly simulate under any input vector sequence. Instead, they rely on statistical information (such as the mean activity and correlations) about the input stream. The pattern dependence problem in these techniques is less severe as these techniques either implicitly consider all input vector sequences or perform a "smoothing" operation on the input streams to reduce dependence on any given sequence.

5.1 Statistical Sampling

Existing techniques for power estimation at the gate and circuit-level can be divided into two classes: *Static* and *dynamic*. Static techniques 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 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. An example of direct simulation techniques is [35].

To address this problem, a *Monte Carlo simulation* technique was proposed in [36]. 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 (see Figure 1).

This approach suffers from three major shortcomings. First, when the vectors are regenerated for simulation, the spatial correlations among various inputs cannot be adequately captured, which may lead to inaccuracy in the power estimates. Second, 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. Finally, there is a general concern about the normality assumption of the sample distribution. Since the stopping criterion is based on such an assumption, if the sample distribution deviates significantly from the normal distribution (which may happen if the number of units per sample is small or the population distribution is ill-behaved), then the simulation may terminate prematurely. Ill-behaved population distributions that may cause premature termination include bi-modal, multi-modal, and distributions with long or asymmetric tails.

The first concern can be addressed by developing a sampling procedure which ran-

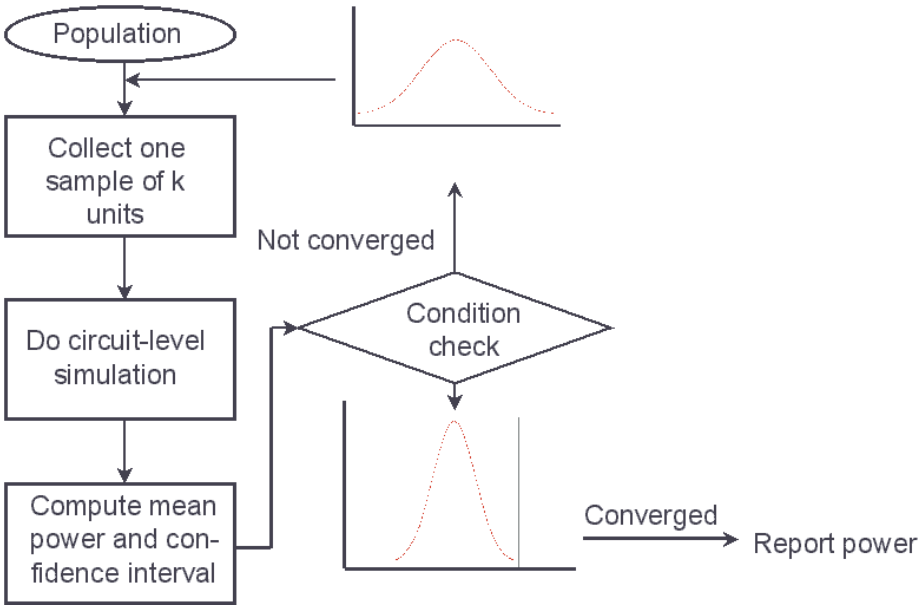


Figure 1: Monte Carlo Simulation Loop.

domly draws the units in the sample from the input population (instead of building an approximate Markov model of the population and then generating samples which conform to this approximate model). Such techniques are known as *simple random sampling* [37]. They may or may not be combined with the Monte Carlo simulation loop.

The second (and to some extent, the third) concern can be addressed by developing more efficient sampling procedures. One such procedure, called *stratified random sampling*, is introduced in [37]. 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 samples are then taken randomly from each stratum. The units in each sample are allocated proportional to the sizes of the strata. This generally results in a significant reduction in the sampling variance (see Figures 2 and 3).

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. The zero-delay power estimates need not to produce high accuracy on a unit-by-unit basis; indeed, they should only show high statistical correlation with circuit-level power estimates (see, for example, the scatter plot for the ISCAS-85 benchmark C1809 shown in Figure 4). When the population size is large, one can use a two-stage stratified sampling procedure to reduce the overhead of predictor calculation and stratification. The proposed two-stage stratified sampling technique can be easily extended to multi-stage sampling.

Compared to [36], the technique of [37] offers the following advantages: 1) It performs sampling directly on the population and thus the estimation results are unbiased; 2) Experimental results on a large set of the ISCAS'85 benchmarks under non-random input

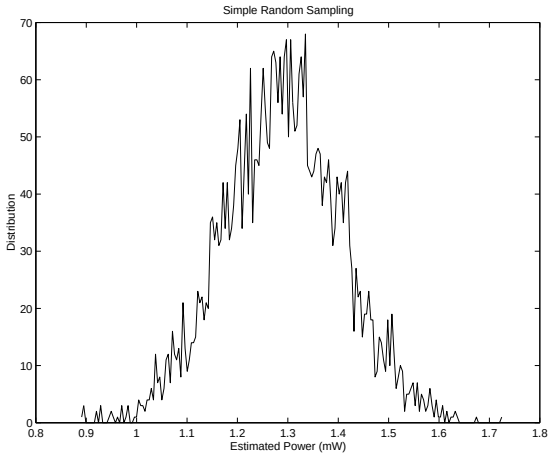


Figure 2: Sample Distribution for Simple Random Sampling.

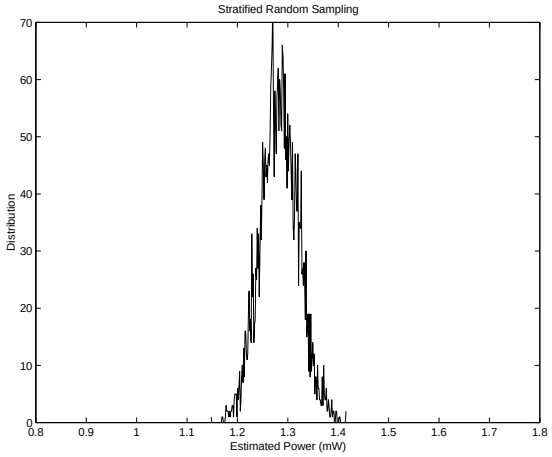


Figure 3: Sample Distribution for Stratified Random Sampling.

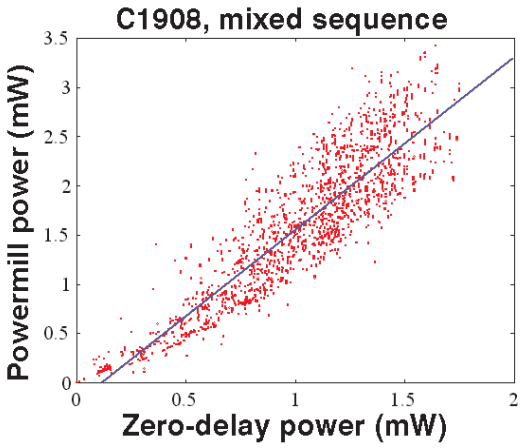


Figure 4: Scatter Plot of Circuit-Level Power versus Zero-Delay Power.

sequences show a 10X improvement in the sampling efficiency; 3) Difficult population distributions can be handled more effectively.

Another efficient sampling procedure, called *linear regression estimation*, is introduced in [34]. The key observation is that the sample variance of the ratio of circuit-level power to zero-delay power tends to be much smaller than that of the circuit-level power by itself. This can be easily seen by examining the scatter plots of circuit-level power versus zero-delay power for a large number of circuits under a variety of input populations.

It is thus more efficient to estimate the mean value of this ratio and then use a regression equation to calculate the mean value of the circuit-level power. Experimental results again show a 10X improvement in the sampling efficiency compared to simple random sampling.

So far we have only considered examples of parametric sampling techniques (i.e., those whose stopping criterion is derived by assuming normality of the sample distribution). A non-parametric sampling technique (i.e., those whose stopping criterion is derived without any a-priori assumption about the sample distribution) is presented in [38]. In general, non-parametric techniques do not suffer from the premature termination problem; however, they tend to be too conservative and lead to over-sampling to achieve the specified error and confidence levels. The technique of [38], which uses the properties of "order statistics", tries to reach a good trade-off between the estimation accuracy and the computational efficiency. More research is needed to assess the efficiency and robustness of non-parametric versus parametric sampling techniques.

For synchronous sequential circuits (e.g., finite state machines driven by a common clock), if we know the state transition graph of the FSM, we can solve the Chapman-Kolmogorov equations for the stationary state probabilities (cf. Section 5.5. Next, based on these probabilities, we can randomly generate a present-state vector which, along with a randomly generated external input vector, determines the next-state vector. With a

second random external input vector (generated according to the statistics of the external input sequence), a unit of the power sample can be constructed. The subsequent units are generated by randomly setting the state line vectors followed by random generation of two external input vectors and power measurement. Unfortunately, the number of Chapman-Kolmogorov equations is exponential in the number of flip-flops, and hence this approach is not possible for large FSMs.

Existing statistical techniques for the estimation of the mean power in synchronous sequential circuits (e.g., finite state machines driven by a common clock) are classified into two groups: Hierarchical and flat. The hierarchical technique performs behavioral or RT-level simulation of the target circuit for all units (i.e., external input patterns) in the user-specified population, and collects the state line values corresponding to each unit. Next, it treats the FSM as a combinational circuit (i.e., it cuts the sequential feedback lines) which receives an extended input sequence and which needs to be simulated at the circuit-level for accurate power estimation; the new sequence is the concatenation of the external input line and the state line values. The FSM power estimation problem is thereby transformed into a sampling problem for the resulting combinational circuit (assuming that all the flip-flops are edge-triggered). The shortcomings of this technique are that it requires RT-level simulation of the target FSM for all units of the population and that the resulting state line values must be stored for sampling in the next phase. The first problem is not critical since the RT-level simulation is orders of magnitude faster than the circuit-level simulation and hence we can afford simulating the whole vector sequence at the RT-level. The second problem may be of concern if the length of the input sequence is large and the computer memory is limited.

The hierarchical technique takes a somewhat more complicated form when the input population is infinite; in this case, the signal and transition probabilities of the state lines must be modeled by a Markov process which is, in turn, derived from a Markov model representation of the external input sequence. This problem has been tackled in [39], where the following two phase procedure is proposed: 1) Use sequence generation based on the given Markov model of each bit of the external input sequence and do Monte Carlo simulation to compute the signal probabilities and transition probabilities of each state line (and hence construct the corresponding Markov model for each state line); 2) Cut the feedback lines in the FSM and do Monte Carlo simulation on the combinational part of the FSM using the bit-level Markov models of the external input and state lines. A major disadvantage of this technique is that in the second step, while estimating power in the combinational circuit, spatial correlations between the signal lines are ignored. This introduces large errors in the power estimates.

The flat technique [36, 40] consists of two phases: Warm-up period and random sampling. The purpose of the warm-up period is to perform a number of input vector simulations to achieve steady state probability conditions on the state lines. Only then, power sampling is performed. This is because, for meaningful power sampling, the state vectors fed into the circuit have to be generated according to their stationary state probability distribution in the FSM. The problems of how to choose an initial state for each sample and how long the warm-up length should be, are discussed in [40]. A

randomness test to dynamically decide the proper warm-up length is proposed in [41]. The test is applied to the sequence of power values observed during the warm-up period (a binning technique is used to transform the sequence of power values into a binary sequence so that the test can be applied). A major concern with this technique is that the randomness test should actually be applied to the sequence of state line patterns, rather than to the sequence of power values. The former task appears to be difficult.

In practice, the warm-up period requires a large number of simulated vectors (depending on the FSM behavior and characteristics of the external input sequence). This makes the efficiency of power estimation for sequential circuits much lower than that for combinational circuits.

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 [42] and the cumulative distribution function for the power dissipation [43]. This information is very useful to chip designers who are interested in reliability analysis and AC/DC noise analysis.

5.2 Probabilistic Compaction

Another approach for reducing the power simulation time is to compact the given long stream of bit vectors using probabilistic automata [44, 45]. The idea in [45] 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. As an example, consider the input sequence shown in Figure 5; the corresponding SSM model is shown in Figure 6, and the compacted sequence is shown in Figure 7.

```
01101010011111010  (bit A)
11100000110111000  (bit B)
10000110101110101  (bit C)
```

Figure 5: Initial Sequence.

The number of states in the probabilistic automaton is proportional to the number of distinct patterns in the initial vector sequence. Since this number may be large (i.e., worst-case exponential in the number of bits in each vector), one has to manage the complexity by either: 1) Partitioning the n bits into b groups with a maximum size of k bits per group and then building a probabilistic automaton for each group of bits independently; 2) Partitioning the long vector sequence into consecutive blocks of

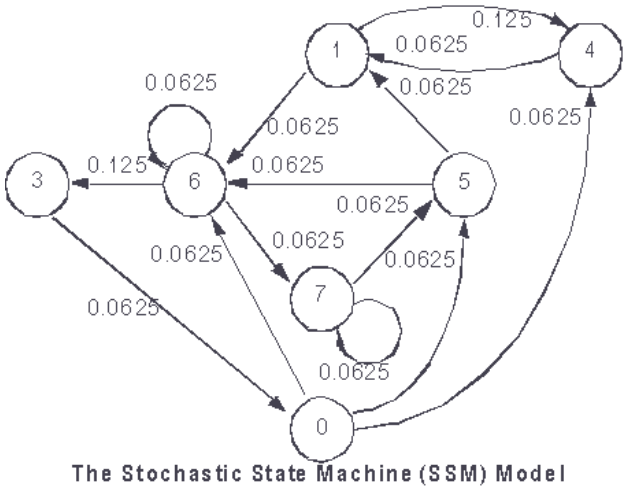


Figure 6: SSM Model for the Sequence.

001010 (bit A)
011000 (bit B)
010001 (bit C)

Figure 7: Compacted Sequence.

vectors such that the number of distinct vectors in each block does not exceed some user defined parameter, say K. The shortcoming of the first approach is that one may generate a bit pattern (vector) that was not present in the initial sequence (since correlations across different groups of bits are ignored). This is, in turn, a problem in certain applications with forbidden input patterns (codes not used, illegal instructions, bad memory addresses, etc.). Thus, the second approach is often more desirable.

An improved algorithm for vector compaction is presented in [46]. 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 [47] as a candidate to solve various data compression problems. This original formulation is extended in [46] 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 [48]. 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. The primary motivation in doing this structuring is to enable a better modeling of the different stochastic levels that are present in sequences that arise in practice.

Another important property of such models is to individualize different operating modes of a circuit or system via higher levels in the hierarchical Markov model, thus providing a high adaptability to different system operating modes (e.g., active, standby, sleep, etc.). The structure of the model itself is general and, with further extensions, can allow an arbitrary number of activations of its sub-models. Once we have constructed the hierarchy for the input sequence, starting with a macro-state, a compaction procedure with a specified compaction ratio is applied to compact the set of micro-states within that macro-state. When done with processing the current macro-state, the control returns to the higher level in the hierarchy and, based on the conditional probabilities that characterize the Markov chain at this level, a new macro-state is entered and the process repeats.

This sequence compaction approach has been extended in [49] to handle FSMs. More precisely, it is shown that: 1) Under the stationarity and ergodicity assumptions, complete capture of the characteristics of the external input sequence feeding into a target FSM is sufficient to correctly characterize the joint (external inputs plus internal states) transition probabilities; 2) If the external input sequence has order k, then a lag-k Markov chain model of this sequence will suffice to exactly model the joint transition probabilities in the target FSM; 3) If the input sequence has order two or higher, then modeling it as a lag-one Markov chain cannot exactly preserve even the first-order joint transition probabilities in the target FSM. The key problem is thus to determine the order of the Markov source which models the external input sequence. In [50], based on the notion of block (metric) entropy, a technique for identifying the order of a *composite* source of information (that is, a source which emits sequences that can be piecewise modeled by Markov chains of different orders) is introduced.

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.

5.3 Probabilistic Simulation

Power dissipation in CMOS circuits comes from three sources: 1) the leakage current, 2) the short-circuit which is due to the DC path between the supply rails during output transitions and 3) the charging and discharging of capacitive loads during logic changes. In a well-designed circuit, the contribution of first two sources is relatively small. Therefore power dissipation at the gate level can be accurately approximated by

only considering the charging and discharging of capacitive loads as given by:

$$Power_{gate} = \frac{1}{2T_c} V_{dd}^2 \sum_n C_n sw_n$$

where T_c is the clock period, V_{dd} is the supply voltage, the summation is performed over all gates (nodes) in the circuit, C_n is the load capacitance of node n and sw_n is the average switching activity of node n (that is, the expected number of signal changes) per clock cycle.

In the above equation, sw_n is related to the timing relationship (or delay) among the input signals of each gate. Indeed, the output signal of a node may change more than once due to unequal signal arriving times at its inputs. The power consumed by these extra signal changes is generally referred as the *glitch power*. If the contribution of the glitch power is relatively small, one can approximate sw_n by using the zero delay model, e. g. assuming that all logic changes propagate through the circuit instantaneously, and hence each node changes logic value at most once. The ratio of the power estimations obtained under a real delay and the zero delay models reflects the significance of glitch power. A typical value for this ratio is 1.20. For some type of circuits, such as parity chains, adders, multipliers, this ratio could be well above 2.0. For this type of circuits, the delay model becomes very important, that is, the delay characterization of each gate must be very accurate. Moreover, CMOS gates have an inertial delay. Only glitches with adequate strength (that is, glitch width) can overcome the gate inertia and propagate through the gate to its output. Accurate modeling of these effects requires significantly longer execution time.

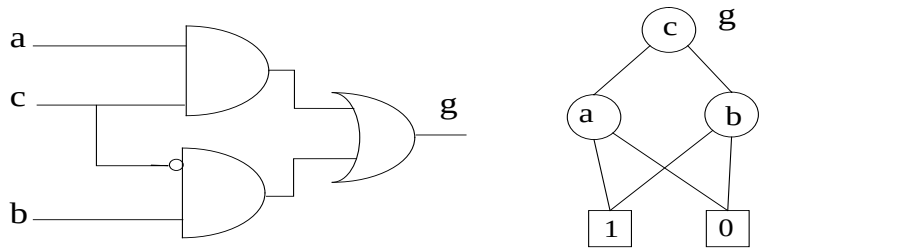
A direct simulative approach to power estimation would enumerate all pairs of input vectors and compute the number of logic changes per vector pair. The "average" switching activity is then the sum over all possible vector pairs of the product of the occurrence probability and the number of the logic changes for each vector pair. This procedure will obviously be exponential in complexity. In the following, we review some important definitions that will help manage this complexity:

1. *Signal probability*: The signal probability $P_s(n)$ of a node n is defined as the average fraction of clock cycle in which the steady state value of n is logic one under the zero delay model.

2. *Transition probability*: The transition $P_t^{ij}(n)$ of a node is defined as the average fraction of clock cycles in which the steady state value of n undergoes a change from logic value i to logic value j under the zero delay model. Note that $P_t^{01}(n) = P_t^{10}(n)$ and $P_t^{01}(n) + P_t^{11}(n) = P_s(n)$.

3. *Temporal independence*: Under temporal independence assumption, the signal value of an input node n at clock cycle i is independent of its signal value at clock cycle $i-1$.

4. *Spatial independence*: Under spatial independence assumption, the logic value of an input node n is independent of the logic value of any other input node m



(a) An example circuit (b) the OBDD representation for g

Figure 8: An OBDD example.

5. *Spatiotemporal correlation*: The spatial and temporal correlation of the input signals are collectively referred as spatiotemporal correlation.

6. *Full signal swing*: All logic transitions are from zero to one and vice versa (that is, no intermediate voltages are present).

The concepts of signal and transition probabilities are very useful for power estimation under the zero delay model. Under the real delay model, the signal and transition probabilities are mainly used to describe the input vector statistics.

Under the zero delay model, $P_t^{01}(n) + P_t^{10}(n)$ can replace the term sw_n in the power calculation equation to give the power consumption for each node. With the temporal independence assumption, $P_t^{01}(n)$ can be written as:

$$P_t^{01}(n) = (1 - P_s(n))P_s(n)$$

Therefore the simulation can be performed using one vector to obtain the signal probabilities instead of using two vectors to explicitly calculate the transition probabilities.

The early works for computing the signal probabilities in a combinational network adopt adopt the spatial independence assumption to keep the problem manageable. In Figure 8, we use signal probability calculation based on OBDDs to illustrate how the spatial independence assumption improves the efficiency. Figure 8 (a) shows the example circuit. Figure 8 (b) shows the global function of node g represented by an OBDD. With the spatial independence assumption, the signal probability $P_s(g)$ can be calculated by recursively cofactoring the global function of g, that is:

$$\begin{aligned} g &= c g_c + \overline{c} g_{\overline{c}} \\ P_s(g) &= P_s(c)P(g_c) + (1 - P_s(c))P(g_{\overline{c}}) \end{aligned}$$

In Figure 8 (b), the left and right branches of the root node compute g_c and $g_{\overline{c}}$, respectively. By recursively applying these equations with respect to all OBDD variables, $P_s(g)$ can be computed efficiently by traversing each OBDD node exactly once using a dynamic programming approach. Without the spatial independence assumption, the

second equation becomes invalid. In the worst case, one may need to explicitly enumerate all the disjoint cubes (paths in the OBDD representation) in function g to compute $P_s(g)$.

When we compute the signal probabilities using OBDD, the OBDDs can be either *global* or *local*. The former refers to OBDDs which are constructed in terms of the variables associated with circuit inputs, while the latter refers to OBDDs which are constructed in terms of the variables associated with some set of intermediate nodes in the fanin cone of the node in question.

We should point out that the temporal and spatial independence assumptions are only made to improve the efficiency at the cost of reducing the accuracy. These assumptions do not hold for most real-life input vectors. Therefore the ability to do power estimation while accounting for real-life temporal and spatial correlations becomes an important criterion when comparing various estimation techniques.

The second class of the switching activity estimation techniques, called static techniques, do not explicitly simulate under the given input vector sequence. This class can be further divided into the following groups: *exact* and *approximate*.

5.4 Exact Probabilistic Analysis

These exact techniques provide the exact switching activity estimates under the assumed delay models and specified input statistics. This is achieved by *implicitly exhaustive enumeration* of all input vectors. Unfortunately, the worst case complexity of these approaches is exponential. Moreover, the data structures employed to perform the implicit enumeration are also exponential in the circuit input size. As a result, the feasibility of these techniques is restricted to small-size circuits.

5.4.1 Exact Techniques under the Zero Delay Model

In [51], each of the circuit inputs is associated with a variable name that represents the input signal probability. Given the algebraic expressions, in term of these variables, of all fanin nodes of a internal circuit node g, an algebraic expression for g can be derived based on the node function of g. The signal probability of each circuit node g is then calculated from the derived algebraic expression. A much more effective approach based OBDDs is proposed in [52]. This latter technique was illustrated with the example of Figure 8. Both of these approaches assume temporal and spatial independence of circuit input signals.

Another OBDD-based approach that considers the temporal correlation of the input signals is proposed in [53]. The technique uses OBDD variables of twice the number of circuit inputs. That is, for each circuit input c, two OBDD variables, c_1 and c_2, are used to represent the signal values of c at time $-\infty$ and 0 in a two-vector simulation. The computation of transition probability of each circuit node g is carried out as follows. Let g^{01} represent the boolean function for g to produce a 0 to 1 signal change under the

zero delay model. We can write:

$$\begin{aligned} g^{01} &= \overline{c_1}\ \overline{c_2} g_{\overline{c_1}\,\overline{c_2}} + \overline{c_1} c_2 g_{\overline{c_1} c_2} + c_1 \overline{c_2} g_{c_1 \overline{c_2}} + c_1 c_2 g_{c_1 c_2} \\ P_t^{01}(g) &= P_t^{00}(c) P(g_{\overline{c_1}\,\overline{c_2}}) + P_t^{01}(c) P(g_{\overline{c_1} c_2}) + P_t^{10}(c) P(g_{c_1 \overline{c_2}}) + P_t^{11}(c) P(g_{c_1 c_2}) \end{aligned}$$

The ordering of the OBDD variables is arranged so that c_1 and c_2 for each circuit input c are next to each other. This ordering provides an effective method to find the cofactors directly from the OBBD nodes. A more efficient method of calculating the transition probabilities without variable duplication is presented in [54].

5.4.2 Exact Techniques under the Real Delay Model

An approach based on *symbolic simulation* is proposed in [55]. In the symbolic simulation, the state of an internal circuit node g is described by a set of symbolic functions defined over time $(-\infty, \infty)$ in the temporal order, e. g. g_{t_0}, g_{t_1}, ...,g_{t_n} where $t_0 = -\infty$ and g_{t_i} specifies the input vectors that will produce logic one over time $[g_{t_i}, g_{t_{i+1}}]$ ($[g_{t_i}, \infty]$) if $i < n$ ($i = n$). The state of each circuit input c is described by two single variable symbolic functions, c_1 and c_2, defined at time $-\infty$ and 0, respectively. The computation of all symbolic functions for each gate can be performed during a topological order. This procedure is similar to event-driven simulation. That is, for each symbolic function defined at time t at any fanin of gate g, a new symbolic function at the output of g at time $t + d$ (d is the delay of the gate) is constructed based on the Boolean function of g and the states of all other fanins of g at time t. The sw_n is calculated by summing the signal probabilities of the exclusive or functions of symbolic functions that are defined at two consecutive time instances, e. g.

$$sw_g = \sum_{i=0}^{n-1} P_s(g_{t_i} \oplus g_{t_{i+1}})$$

The major disadvantage of this method is that it is computationally intractable, as the number of symbolic functions for each gate could be large. Moreover the symbolic functions could become very complicated when one attempts to model the glitch propagation mechanism accurately and potentially filter out glitches that have short width.

So far no exact techniques have been proposed for considering the spatial correlation of input signals under either the zero or the real delay model. The difficulty lies in the fact that the spatial correlations can in general exist among every m-tuple of inputs where m ranges from 2 to the number of circuit inputs n. In the worst case, there is an exponential number of spatial correlation parameters that need to be considered compared to the $4n$ for the temporal correlation case.

5.5 Approximate Techniques

These techniques are developed as approximation techniques for the implicit enumeration approaches. They are referred as approximate probabilistic techniques mainly because the probabilistic quantities such as signal and transition probabilities are explicitly (vs. implicitly in exact techniques) propagated in the networks.

5.5.1 Probabilistic Techniques under the Zero Delay Model

An efficient algorithm to estimate the signal probability of each internal node using pair-wise correlation coefficients among circuit nodes is proposed in [56]. This technique allows the spatial correlations between pairs of circuit input signals to be considered. A more general approach that accounts for spatiotemporal correlations, is proposed in [54]. The mathematical foundation of this extension is a four state time-homogeneous Markov chain where each state represents some assignment of binary values to two lines x and y and each edge describes the conditional probability for going from one state to next. The computational requirement of this extension can however be high since it is linear in the product of the number of nodes and number of paths in the OBDD representation of the Boolean function in question. A practical method using local OBDD constructions and dynamic level bounding is described by the authors.

This work has been extended to handle highly correlated input streams using the notions of conditional independence and isotropy of signals [57]. Based on these notions, it is shown that the relative error in calculating the signal probability of a logic gate using pairwise correlation coefficients can be bounded from above.

5.5.2 Probabilistic Techniques under the Real Delay Model

In [58] ($CREST$), the concept of *probability waveforms* is proposed to estimate the mean and variance of the current drawn by each circuit node. Although $CREST$ was originally developed for switch level simulation, the concept of probability waveforms can be easily extended to the gate level as well. A probability waveform consists of an initial signal probability and a sequence of transition events occurring at different time instances. Associated with each transition event is the probability of the signal change. In a probability waveform one can calculate the signal probability at any time instance t from the initial signal probability and probabilities of each transition event which occurred before t. The propagation mechanism for probability waveforms is event-driven in nature. For each transition event arrived at time t at the input of a gate g, a transition event is scheduled at the gate output at time $t + d$ (d is the delay of the gate); probability of the event is calculated based on the probability of the input events and the signal probabilities of other fanins at time t.

In [59] ($DENSIM$) the notion of *transition density* which is the average number of transitions per second is proposed. It can replace SW_n/T_c in power calculation equation to compute the power dissipation for each node. An efficient algorithm based on Boolean difference operation is proposed to propagate the transition densities from circuit inputs throughout the circuit with the transition density of each node calculated based on the following formula:

$$D(y) = \sum_{i=1}^{n} P(\frac{\partial y}{\partial x_i}) D(x_i)$$

where y is the output of a node, $x_i's$ are the inputs of y, $D(y)$ is the transition density of node y and $\frac{\partial y}{\partial x_i}$ is the Boolean difference of y respect to x_i. $P(\frac{\partial y}{\partial x_i})$ is calculated

using OBDDs. The accuracy of the transition density equation can be improved by considering higher order Boolean difference [60].

Under the real delay model, the correlation between gate fanins is the major source of inaccuracy for the probabilistic techniques. Moreover, because the signals may change more than once, it is very difficult to accurately model these correlations. In [61], based on an assumption on the probability distribution function of the glitch width, a conceptual low-pass filter module is proposed that improves the accuracy of $DENSIM$.

In [62] (TPS), the concept of probability waveforms is extended to partially account for the signal correlation among different gate fanins. The signal correlation is approximated by the steady-state correlations under the zero delay model. To efficiently implement the techniques, the probability waveform of a node is divided into four *tagged waveforms* based on initial and final steady state. For an 2-input node g, there are 16 joint tagged waveforms at the gate inputs. After the probability waveforms for all 16 joint tagged waveforms are calculated, they are combined into four tagged waveforms according to a forcing set table [62] derived from the node function of g. The OBDDs are used to calculate the probabilities of each tagged waveform and the signal correlations. This approach requires significantly less memory and runs much faster than symbolic simulation, yet achieves high accuracy, e.g., the average error in aggregate power consumption is about 10%. One interesting characteristic of this approach is that it will give exact power estimate for the zero delay model if the zero delay model is assumed. Moreover the technique can be combined with [54, 57] to consider the spatiotemporal correlations of circuit input signals.

Both $DENSIM$ and TPS use OBDDs to improve their efficiency; their computational work can be divided into two phases. The first phase constructs the required OBDDs and computes the signal probabilities or correlations using OBDDs; the second phase computes either the transition density or the tagged waveforms during a post-order traversal from circuit inputs to circuit outputs.

5.5.3 Probabilistic Techniques for Finite State Machines

The abovementioned probabilistic methods for power estimation focus on combinational logic circuits. Accurate average switching activity estimation for FSMs is considerably more difficult than that for combinational circuits for two reasons:

- The probability of the circuit being in each of its possible states has to be calculated
- The present-state line inputs of the FSM are strongly correlated (that is, they are temporally correlated due to the machine behavior as represented in its State Transition Graph description and they are spatially correlated because of the given state encoding).

A first attempt at estimating switching activity in FSMs was presented in [55]. The idea is to "unroll" the next-state logic once (thus capturing the temporal correlations of present-state lines) and then perform symbolic simulation on the resulting circuit (which

is hence treated as a combinational circuit). This method does not however capture the spatial correlations among present-state lines and makes the simplistic assumption that the state probabilities are uniform.

The above work was improved on in [63] as follows. For each state $s_i,\ 1 \leq i \leq K$ in the STG, we associate a variable $prob(s_i)$ corresponding to the steady-state probability of the machine being in state s_i at $t = \infty$. For each edge e in the STG, we have $e.Current$ signifying the state that the edge fans out from, $e.Next$ signifying the state that the edge fans out to, and $e.Input$ signifying the input combination corresponding to the edge. Given static probabilities for the primary inputs to the machine, we can compute $prob(Input)$, the probability of the combination $Input$ occurring. [1] We can compute $prob(e.Input)$ using:

$$prob(e.Input) \ = \ prob(e.Current) \times prob(Input)$$

For each state s_i we can write an equation:

$$prob(s_i) \ = \sum_{\forall\ e\ such\ that\ e.Next\ =\ s_i} prob(e.Input)$$

Given K states, we obtain K equations out of which any one equation can be derived from the remaining $K-1$ equations. We have a final equation:

$$\sum_{i=1}^{K} prob(s_i) \ = \ 1$$

This linear set of K equations can be solved to obtain the different $prob(s_i)$'s. This system of equations is known as the Chapman-Kolmogorov equations for a discrete-time discrete-transition Markov process. Indeed, if the process satisfies the conditions that it has a finite number of states, its essential states form a single-chain and it contains no periodic-states, then the above system of equations will have a unique solution.

The Chapman-Kolmogorov method requires the solution of a linear system of equations of size 2N, where N is the number of flip-flops in the machine. In general, his method cannot handle circuits with large number of flip-flops because it requires explicit consideration of each state in the circuit. On the positive side, state probabilities for some very large FSMs have been calculated using a fully implicit technique described in [64].

The authors of [63] also describe a method for approximate switching activity estimation of sequential circuits. The basic computation step is the solution of a non-linear system of equations as follows:

$$prob(ns_1) \ = \ prob(f_1(i_1,\ i_2,\ \cdots,\ i_M,\ ps_1,\ ps_2,\ \cdots,\ ps_N))$$

$$prob(ns_2) \ = \ prob(f_2(i_1,\ i_2,\ \cdots,\ i_M,\ ps_1,\ ps_2,\ \cdots,\ ps_N))$$

[1] Static probabilities can be computed from specified transition probabilities.

$$\cdots$$

$$prob(ns_N) \;=\; prob(f_N(i_1,\; i_2,\; \cdots,\; i_M,\; ps_1,\; ps_2,\; \cdots,\; ps_N))$$

where $prob(ns_i)$ corresponds to the probability that ns_i is a 1, and $prob(f_i(i_1,\; i_2,\; \cdots,\; i_M,\; ps_1,\; ps_2,\; \cdots,\; ps_N))$ corresponds to the probability that $f_i(i_1,\; i_2,\; \cdots,\; i_M,\; ps_1,\; ps_2,\; \cdots,\; ps_N)$ is a 1, which is of course dependent on the $prob(ps_j)$ and the $prob(i_k)$.

We are interested in the steady state probabilities of the present and next-state lines implying that:

$$prob(ps_i) \;=\; prob(ns_i) \;=\; p_i \quad 1 \leq i \leq N$$

A similar relationship was used in the Chapman-Kolmogorov equations.

The set of equations given the values of $prob(i_k)$ becomes:

$$y_1 = p_1 - g_1(p_1,\; p_2,\; \cdots,\; p_N) \;=\; 0$$

$$y_2 = p_2 - g_2(p_1,\; p_2,\; \cdots,\; p_N) \;=\; 0$$

$$\cdots$$

$$y_N = p_N - g_N(p_1,\; p_2,\; \cdots,\; p_N) \;=\; 0 \qquad (1)$$

where the g_i's are non-linear functions of the p_i's. We will denote the above equations as $Y(P) = 0$ or as $P = G(P)$. In general the Boolean function f_i can be written as a list of minterms over the i_k and ps_j and the corresponding g_i function can be easily derived.

The fixed point (or zero) of this system of equations $P = G(P)$ (or $Y(P) = 0$) can be found using the Picard-Peano (or Newton-Raphson) iteration [65]. The uniqueness or the existence of the solution is not guaranteed for an arbitrary system of non-linear equations. However, since in our application we have a correspondence between the non-linear system of equations and the State Transition Graph of the sequential circuit, there will exist at least one solution to the non-linear system. Further, convergence is guaranteed under mild assumptions for our application.

Increasing the number of variables or the number of equations in the above system results in increased accuracy [148]. For a wide variety of examples, it is shown that the approximation scheme is within 1-3but is orders of magnitude faster for large circuits. Previous sequential switching activity estimation methods exhibit significantly greater inaccuracies.

6 Transistor-Level Power Estimation

Transistor-level simulators provide the highest accuracy for circuit power estimation. They are capable of handling various device models, different circuit design styles, single and multi-phase clocking methodologies, tristate drives, etc. However, they suffer from memory and execution time constraints and are not suitable for large, cell-based designs. Circuit techniques for power measurement (capacitive and short-circuit power components) using the "power meters" is described in [66, 67]. A fast and accurate

circuit-level simulator based on the stepwise equivalent conductance and piecewise linear waveform approximation has been described in [68].

PowerMill [69] is a transistor-level power simulator and analyzer which applies an event-driven timing simulation algorithm (based on simplified table-driven device models, circuit partitioning and single-step nonlinear iteration) to increase the speed by two to three orders of magnitude over SPICE while maintaining an accuracy of within 10power information (instantaneous, average and RMS current values) as well as the total power consumption (due to capacitance currents, transient short circuit currents, and leakage currents).

7 Conclusions

The increased degree of automation of industrial design frameworks has produced a substantial change in the way digital ICs are developed. The design of modern systems usually starts from specifications given at a very high level of abstraction. This is because existing EDA tools are able to automatically produce low-level design implementations directly from descriptions of this type.

It is widely recognized that power consumption has become a critical issue in the development of digital systems; then, electronic designers need tools that allow them to explicitly control the power budget during the various phases of the design process. This is because the power savings obtainable through automatic optimization are usually more significant than those achievable by means of technological choices (e.g., process and supply-voltage scaling).

In this paper, we have provided a non-exhaustive review of existing methodologies and tools for high-level power modeling and estimation, as well as for power-constrained synthesis and optimization. Such methodologies and tools 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 is in view of technological and strategic changes in the world of microelectronics.

References

[1] F. N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Transactions on VLSI Systems*, Vol. 2, No. 4, pp. 446-455, 1994.

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

[3] J. M. Rabaey and M. Pedram Editors *Low Power Design Methodologies*, Kluwer Academic Publishers, 1996.

[4] J. Mermet and W. Nebel Editors, *Low Power Design in Deep Submicron Electronics*, Kluwer Academic Publishers, 1997.

[5] T. Sato, Y. Ootaguro, M. Nagamatsu, H. Tago, "Evaluation of Architectural-Level Power Estimation for CMOS RISC Processors," *ISLPE-95: IEEE International Symposium on Low Power Electronics*, pp. 44-45, San Jose, CA, October 1995.

[6] C.-L. Su, C.-Y. Tsui, A. M. Despain, "Low Power Architecture Design and Compilation Techniques for High-Performance Processors," *IEEE CompCon-94*, pp. 489-498, February 1994.

[7] V. Tiwari, S. Malik, A. Wolfe, "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization," *IEEE Transactions on VLSI Systems*, Vol. 2, No. 4, pp. 437-445, 1994.

[8] C-T. Hsieh, M. Pedram, H. Mehta, F. Rastgar, "Profile-Driven Program Synthesis for Evaluation of System Power Dissipation," *DAC-34: ACM/IEEE Design Automation Conference*, pp. 576-581, Anaheim, CA, June 1997.

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

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

[11] K. T. Cheng, V. D. Agrawal, "An Entropy Measure for the Complexity of Multi-Output Boolean Functions," *DAC-27: ACM/IEEE Design Automation Conference*, pp. 302-305, Orlando, FL, June 1990.

[12] F. Ferrandi, F. Fummi. E. Macii, M. Poncino, D. Sciuto, "Power Estimation of Behavioral Descriptions," *DATE-98: IEEE Design Automation and Test in Europe*, pp. 762-766, Paris, France, February 1998.

[13] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on CAD*, pp. 667-691, August 1986.

[14] A. Tyagi, "Entropic Bounds on FSM Switching," *IEEE Transactions on VLSI Systems*, Vol. 5, No. 4, pp. 456-464, 1997.

[15] D. Marculescu, R. Marculescu and M. Pedram, "Theoretical bounds for switching activity analysis in finite-state machines," *ISLPED-98: ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 36-41, Monterey, CA, August 1998.

[16] K. Muller-Glaser, K. Kirsch, K. Neusinger, "Estimating Essential Design Characteristics to Support Project Planning for ASIC Design Management," *ICCAD-91: IEEE/ACM International Conference on Computer Aided Design*, pp. 148-151, Santa Clara, CA, November 1991.

[17] M. Nemani, F. Najm, "High-Level Area Prediction for Power Estimation," *CICC-97: Custom Integrated Circuits Conference*, pp. 483-486, Santa Clara, CA, May 1997.

[18] M. Nemani, F. Najm, "High-Level Area and Power Estimation for VLSI Circuits," *ICCAD-97: IEEE/ACM International Conference on Computer Aided Design*, pp. 114-119, San Jose, CA, November 1997.

[19] P. Landman, J. Rabaey, "Activity-Sensitive Architectural Power Analysis for the Control Path," *ISLPD-95: ACM/IEEE International Symposium on Low Power Design*, pp. 93-98, Dana Point, CA, April 1995.

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

[21] J. M. Chang, M. Pedram, "Module Assignment for Low Power," *EuroDAC-96: IEEE European Design Automation Conference*, pp. 376-381, Geneva, Switzerland, September 1996.

[22] 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.

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

[24] L. Benini, A. Bogliolo, M. Favalli, G. De Micheli, "Regression Models for Behavioral Power Estimation," *PATMOS-96: International Workshop on Power and Timing Modeling, Optimization and Simulation*, pp. 179-186, Bologna, Italy, September 1996.

[25] L. Benini, A. Bogliolo, G. De Micheli, "Characterization-Free Behavioral Power Modeling," *DATE-98: IEEE Design Automation and Test in Europe*, pp. 767-773, Paris, France, February 1998.

[26] L. Benini, A. Bogliolo, G. De Micheli, "Adaptive Least Mean Square Behavioral Power Modeling," *EDTC-97: IEEE European Design and Test Conference*, pp. 404-410, Paris, France, March 1997.

[27] 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.

[28] P. Landman, J. Rabaey, "Power Estimation for High-Level Synthesis," *EDAC-93: IEEE European Conference on Design Automation*, pp. 361-366, Paris, France, February 1993.

[29] S. Gupta, F. N. Najm, "Power Macromodeling for High-Level Power Estimation," *DAC-34: ACM/IEEE Design Automation Conference*, pp. 365-370, Anaheim, CA, June 1997.

[30] 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.

[31] H. Mehta, R. Owens, M. J. Irwin, "Energy Characterization Based on Clustering," *DAC-33: ACM/IEEE Design Automation Conference*, pp. 702-707, Las Vegas, NV, June 1996.

[32] Q. Wu, C-S. Ding, C-T. Hsieh, M. Pedram, "Statistical Design of Macro-Models for RT-Level Power Evaluation," *ASPDAC-2: ACM/IEEE Asia South Pacific Design Automation Conference*, pp. 523-528, Chiba, Japan, January 1997.

[33] Q. Qiu, Q. Wu, M. Pedram, C-S. Ding, "Cycle-Accurate Macro-Models for RT-Level Power Analysis," *ISLPED-97: ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 125-130, Monterey, CA, August 1997.

[34] C-T. Hsieh, C-S. Ding, Q. Wu, M. Pedram, "Statistical Sampling and Regression Estimation in Power Macro-Modeling," *ICCAD-96: IEEE/ACM International Conference on Computer Aided Design*, pp. 583-588, San Jose, CA, November 1996.

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

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

[37] C-S. Ding, C-T. Hsieh, Q. Wu, M. Pedram, "Stratified Random Sampling for Power Estimation," *ICCAD-96: IEEE/ACM International Conference on Computer Aided Design*, San Jose, CA, pp. 577-582, November 1996.

[38] L-P. Yuan, C-C. Teng, S-M. Kang, "Statistical Estimation of Average Power Dissipation in CMOS VLSI Circuits Using Nonparametric Technique," *ISLPED-96: ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 73-78, Monterey, CA, August 1996.

[39] F. N. Najm, S. Goel, and I. N. Hajj, "Power estimation in sequential circuits" *DAC-32: ACM/IEEE Design Automation Conference*, pp. 635-640, San Francisco, CA, June 1995.

[40] T-L. Chou, K. Roy, "Statistical Estimation of Sequential Circuit Activity," *ICCAD-95: IEEE/ACM International Conference on Computer Aided Design*, pp. 34-37, San Jose, CA, November 1995.

[41] L-P. Yuan, C-C. Teng, S-M. Kang, "Statistical Estimation of Average Power Dissipation in Sequential Circuits," *DAC-33: ACM/IEEE Design Automation Conference*, pp. 377-382, Anaheim, CA, June 1996.

[42] A. Hill, C-C. Teng, S. M. Kang, "Simulation-Based Maximum Power Estimation," *ISCAS-96: IEEE International Symposium on Circuits and Systems*, Vol. IV, pp. 13-16, Atlanta, GA, May 1996.

[43] C-S. Ding, Q. Wu, C-T. Hsieh, M. Pedram, "Statistical Estimation of the Cumulative Distribution Function for Power Dissipation in VLSI Circuits," *DAC-34: ACM/IEEE Design Automation Conference*, pp. 371-376, Anaheim, CA, Jun. 1997.

[44] C-Y. Tsui, D. Marculescu, R. Marculescu and M. Pedram, "Improving the Efficiency of Power Simulators by Input Vector Compaction," *DAC-33: ACM/IEEE Design Automation Conference*, pp. 165-168, Las Vegas, NV, Jun. 1996.

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

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

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

[48] R. Marculescu, D. Marculescu, M. Pedram, "Power Estimation Using Hierarchical Markov Models," *DAC-34: ACM/IEEE Design Automation Conference*, pp. 570-575, Anaheim, CA, Jun. 1997.

[49] D. Marculescu, R. Marculescu, M. Pedram, "Sequence Compaction for Probabilistic Analysis of Finite State Machines," *DAC-34: ACM/IEEE Design Automation Conference*, pp. 12-15, Anaheim, CA, Jun. 1997.

[50] R. Marculescu, D. Marculescu, M. Pedram, "Composite Sequence Compaction for Finite State Machines Using Block Entropy and Higher-Order Markov Models," *ISLPED-97: ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 190-195, Monterey, CA, Aug. 1997.

[51] K. P. Parker and J. McCluskey, "Probabilistic treatment of general combinational networks," *IEEE Transactions on Computers*, Vol C24, pp. 668-670, June 1975.

[52] S. Chakravarty, "On the Complexity of Using BDDs for the Synthesis and Analysis of Boolean Circuits," *27th Annual Allerton Conference on Communication, Control and Computing*, pages 730-739, 1989.

[53] P. Schneider and U. Schlichtmann, "Decomposition of Boolean Functions for Low Power Based on a New Power Estimation Technique," *WLPD-94: International Workshop on Low Power Design*, pp. 123-128, Napa, CA, April 1994.

[54] R. Marculescu, D. Marculescu and M. Pedram, "Switching Activity Analysis Considering Spatiotemporal Correlation," *ICCAD-94: IEEE/ACM International Conference on Computer Aided Design*, pp. 294-299, San Jose, CA, November 1994.

[55] A. Ghosh, S. Devadas, K. Keutzer, S. Malik and J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits," *DAC-92: ACM/IEEE Design Automation Conference*, pp. 253-259, Anaheim, CA, June 1992.

[56] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco, "Testability Measures in Pseudorandom Testing," *IEEE Transactions on CAD*, Vol. 11, pp. 794-800, June 1992.

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

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

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

[60] T. L. Chou, K. Roy and S. Prasad, "Estimation of Circuit Activity Considering Signal Correlation and Simultaneous Switching," *ICCAD-94: IEEE/ACM International Conference on Computer Aided Design*, pp. 300-303, San Jose, CA, November 1994.

[61] F. Najm, "Low-Pass Filter for Computing the Transition Density in Digital Circuits," *IEEE Transactions on CAD*, Vol. 13, No. 9, pp. 1123-1131, September 1994.

[62] C-Y. Tsui, M. Pedram, A. M. Despain, "Efficient Estimation of Dynamic Power Dissipation Under a Real Delay Model," *ICCAD-93: IEEE/ACM International Conference on Computer Aided Design*, pp. 224-228, Santa Clara, CA, November 1993.

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

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

[65] H. M. Lieberstein. *A Course in Numerical Analysis*. Harper & Row Publishers, 1968.

[66] S. M. Kang, "Accurate Simulation of Power Dissipation in VLSI Circuits" *IEEE J. Solid-State Circuits*, Vol. 21, pp. 889-891, October 1986.

[67] G. Y. Yacoub, W. H. Ku, "An Enhanced Technique for Simulating Short-Circuit Power Dissipation," *IEEE Journal of Solid-state Circuits*, Vol. 24, pp. 844-847, June 1989.

[68] P. Buch, S. Lin, V. Nagasamy and E. S. Kuh, "Techniques for Fast Circuit Simulation Applied to Power Estimation of CMOS circuits," *SLPD-95: ACM/IEEE International Symposium on Low Power Design,* pp. 135-138, Dana Point, CA, April 1995.

[69] C. X. Huang, B. Zhang, A-C. Deng and B. Swirski, "The Design and Implementation of PowerMill," *SLPD-95: ACM/IEEE International Symposium on Low Power Design,* pp. 105-110, Dana Point, CA, April 1995.