Power Estimation Techniques for Integrated Circuits

Farid N. Najm

ECE Dept. and Coordinated Science Lab.
University of Illinois at Urbana-Champaign

Abstract

With the advent of portable and high-density microelectronic devices, the power dissipation of very large scale integrated (VLSI) circuits is becoming a critical concern. Accurate and efficient power estimation during the design phase is required in order to meet the power specifications without a costly redesign process. Recently, a variety of power estimation techniques have been proposed, most of which are based on: 1) the use of simplified delay models, and 2) modeling the long-term behavior of logic signals with probabilities. The array of available techniques differ in subtle ways in the assumptions that they make, the accuracy that they provide, and the kinds of circuits that they apply to. In this tutorial, I will survey the many power estimation techniques that have been recently proposed and, in an attempt to make sense of all the variety, I will try to explain the different assumptions on which these techniques are based, and the impact of these assumptions on their accuracy and speed.

1. Introduction

The continuing decrease in feature size and the corresponding increase in chip density and operating frequency have made power consumption a major concern in VLSI design [1, 2]. Modern microprocessors are indeed *hot*, with typical power dissipation values ranging from 8 Watts to 60 Watts, for large chips. Excessive power dissipation in integrated circuits is undesirable for two reasons: 1) High power dissipation causes overheating, which degrades performance and reduces chip lifetime. To control their temperature levels, high power chips require specialized and costly packaging and heat-sink arrangements. 2) The demand for portable electronics has created a need for very low-power chips to help prolong the battery life of portable equipment. Thus, there is a need to limit the power consumption in many chip designs. Indeed, the Semiconductor Industry Association has identified low-power design techniques as a *critical technological need* [3].

Managing the power of an Integrated Circuit (IC) design adds to a growing list of problems that IC designers have to contend with. Computer-Aided Design (CAD) tools are needed to help with the power management tasks. Indeed, the overall design methodology needs to be modified to account for power during the design process by helping designers to make trade-offs that reduce the power dissipation. In the same way that testability became an up-front design concern in the 80s, power is now the up-front design concern in the 90s. In the same way that scan design became part of mainstream design methodologies to guarantee testability, we now need general, easy to apply, automatic design techniques for low-power design.

To address these needs, many researchers have responded in various ways, such as by proposing power estimation techniques, low-power library development, low-power optimization techniques, and low-power synthesis tools.

Power estimation is needed at different points in the design process. Ideally, one would like to estimate the power of the design very early on, such as when only a high level (behavioral) description of the design is available. Such a capability would save precious design time and would provide designers with power estimation at a time when the design is still sufficiently flexible that major design changes can be made rather cheaply. I refer to this as *high-level* power estimation. While power estimation from a truly behavioral description is not feasible today, some techniques have been proposed that work with a (moderately) high-level design description. Specifically, some proposed techniques work at the Register Transfer Level (RTL), i.e., when the circuit is described in terms of memory elements and combinational *black boxes* (described only with Boolean equations).

While it is highly desirable, high-level power estimation is also inevitably inaccurate, or approximate. Thus, it is also important to accurately estimate the power once the low-level details of the circuit become available, such as its gate-level or switch-level description. This *low-level* power estimation problem has received much more attention in the literature, and many estimation techniques at this level have been proposed.

After a more detailed description of the power estimation problem, in the next section, the rest of the paper will provide a discussion of many recently proposed estimation techniques.

2. Detailed problem description

By *power estimation* I will generally refer to the problem of estimating the *average* power dissipation of a digital circuit. This is different from estimating the *worst case* instantaneous power [4–6, 10], also referred to as the *voltage drop problem*. Another related problem is that of providing an upper bound on the average power without necessarily bounding the instantaneous power [20]. These techniques will not be discussed in this paper. Instead, I will focus on average power estimation, which is directly related to chip heating and temperature and to battery lifetime.

A simple and straight-forward method of average power estimation is to *simulate* the circuit, say using a circuit simulator, to obtain the power supply voltage and current waveforms, from which the average power can be computed. Techniques of this kind were the first to be proposed [11, 12]. Since they are based on circuit simulation, these techniques can be quite expensive. In order to improve computational efficiency, several other simulation-based techniques were also proposed using various kinds of RTL, gate-, switch-, and circuit-level simulation [13–18, 46–50]. Given a set of input patterns or waveforms, the circuit is simulated, and a power value is reported based on the simulation results. Almost all of these techniques assume that the supply and ground voltages are fixed, and only the supply current waveform is estimated.

Even though these simulation-based techniques can be efficient, their utility in practice is limited because the estimate of the power which they provide corresponds directly to the input patterns that were used to drive the simulation. This points to the central problem in power estimation, namely that the power dissipation is *input pattern-dependent.* Indeed, in most modern logic styles, the chip components (gates, cells) draw power supply current only during a logic transition (if we ignore the small leakage current). While this is considered an attractive low-power feature of these technologies, it makes the power-dissipation highly dependent on the *switching activity* inside these circuits. Simply put, a more active circuit will consume more power. Since internal activity is determined by the input signals, then the circuit power is input pattern-dependent.

In practice, the pattern-dependence problem is a serious limitation. Often, the power dissipation of a circuit block may need to be estimated when the rest of the chip has not yet been designed, or even completely specified. In such a case, very little may be known about the inputs to this block, and exact information about its inputs would be impossible to obtain. Furthermore, for a microprocessor or a Digital Signal Processing (DSP) chip, the exact *data inputs* can not be determined a priori, because they depend on how the chip is deployed in the field.

Recently, several techniques have been proposed to overcome this problem by using *probabilities* as a compact way to describe a large set of possible logic signals, and then studying the power resulting from the collective influence of all these signals. In order to use these techniques, the user only specifies *typical* behavior at the circuit inputs, in the form of transition probability, or average frequency. If typical input pattern sets are available, then the required input probability or frequency information can be easily obtained by a simple averaging procedure. The rest of this paper is devoted to discussing these techniques.

I will classify power estimation techniques as being either *probabilistic* or *statistical.* I call an approach *probabilistic* when it is based on propagating a probability measure directly through the logic. To perform this, special models for circuit blocks (gates) must be developed and stored in the cell library. In contrast, other techniques, that I will refer to as *statistical*, do not require specialized circuit models. Instead, they use traditional simulation models and simulate the circuit, using existing simulation capabilities, for a limited number of *randomly generated* input vectors while monitoring the power. These vectors are generated from user-specified probability information about the circuit inputs. Essentially, these techniques are based on statistical mean estimation resulting from a Monte Carlo procedure. Using statistical estimation techniques, one can determine when to stop the simulation in order to obtain certain user-specified accuracy and confidence.

In the next section, I will discuss power estimation techniques that operate at the gate level, while section 4 presents techniques that work at a higher level of abstraction. Whenever possible, I will comment on the accuracy and speed of the different approaches. However, accuracy comparisons are often hard to do because the published techniques were not all tested on a common set of benchmark designs.

3. Low-level power estimation

Most techniques in this class simplify the problem by assuming a simplified circuit model, as follows:

(1) It is assumed that the power supply and ground voltage levels throughout the chip are fixed.
(2) It is assumed that the circuit is built of CMOS logic gates and edge-triggered flip-flops (FFs), and is synchronous, as shown in Fig. 1.
(3) Only the charging/discharging current is considered, so that the short-circuit current during switching is neglected [7].

Therefore, the average power dissipation of a circuit can be broken down into (a) the power consumed by the flip-flops and (b) that consumed by the combinational logic blocks. Correspondingly, one way to estimate the power is to use the following two-step approach:

1. Solve for the FF power by examining the behavior of the whole circuit as a finite state machine (FSM), and measure statistics of the FF outputs.
2. Use the statistics at the FF outputs, resulting from the FSM analysis, to compute the power for the combinational circuit block.

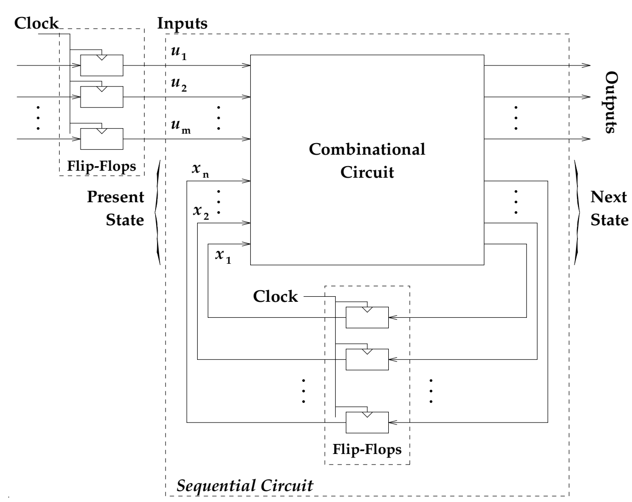

Figure 1. A combinational circuit embedded in a synchronous sequential design.

This process is easily formulated using probabilities, by defining probability measures that characterize the transitions made by a logic signal. We start with the following two:

Definition 1. (signal probability): *The signal probability $P_s(x)$ at a node x is defined as the average fraction of clock cycles in which the steady state value of x is a logic high.*

Definition 2. (transition probability): *The transition probability $P_t(x)$ at a node x is defined as the average fraction of clock cycles in which the value of x at the end of the cycle is different from its initial value.*

The signal probability is a relatively old concept that was first introduced to study circuit testability [9]. In what follows, we will see how probabilities are relevant to power estimation as we consider separately the computation of the FF power and the combinational circuit power.

3.1. Flip-flop power

Whenever the clock triggers the FFs, some of them will make a transition and will draw power. Thus FF power is drawn in synchrony with the clock. If the transition probabilities $P_t(x)$ at the FF outputs are known, then the average power consumed by one flip-flop is simply:

$$\frac{1}{2T_c} V_{dd}^2 C_x P_t(x)$$

where T_c is the clock period and C_x is the total capacitance at the FF output.

Thus the computation of the FF power reduces to finding the FF transition probabilities. However, computing the probabilities $P_t(x_i)$ from the FSM input signal and/or transition probabilities is not trivial. In fact, it can be shown that finding these probabilities *exactly* is $\mathcal{NP}$-hard. Even finding them *approximately* is not easy, because the feedback creates the difficult situation where future signal values are related to their past and present values.

3.1.1. Probabilistic techniques

In trying to compute the probabilities at the state bits of the FSM, it is tempting to consider finding the *state occupation probability*, for every state in the FSM's state transition graph (STG), or the *state transition probability* associated with every edge in the STG. This, however, gets very expensive due to the exponential explosion in the number of states, even for FSMs of moderate size. One technique, given in [28], completely ignores this problem and assumes that all states (of the FSM) are equally probable, which is not true in practice.

Other techniques [40–43] have been proposed that are based on the simplifying assumption that the FSM is *Markov* [34] (so that its future is independent of its past once its present state is specified). This assumption is somewhat restrictive because it is only true when the sequence of input vectors at the FSM primary inputs are independent. Some of these techniques compute only the probabilities (signal and transition) at the FF outputs, while others also compute the power. The approach in [40] solves directly for the transition probabilities on the present state lines using the Chapman-Kolmogorov equations [33, 34], which is computationally too expensive. Another approach that also attempts a direct solution of the Chapman-Kolmogorov equations was given in [41]. While it is more efficient, it remains quite expensive, so that the largest test case presented contains less than 30 FFs.

Better solutions are offered by two recent papers [42, 43], which assume the FSM primary inputs are independent, and which are based on solving a non-linear system that gives the present state line probabilities, as follows. Let a vector of present state signal probabilities $P_{in} = [p_1, p_2, \ldots, p_n]$ be applied to the combinational logic block *and* let the n present state signals be *independent*. At the outputs of the combinational logic, let the corresponding next state node probability vector be P_{out}. The mapping from P_{in} to P_{out} is some non-linear function that is determined by the Boolean function implemented by the logic. We denote this vector-valued function by $F(\cdot)$, so that $P_{out} = F(P_{in})$ (assuming, for now, the FSM primary input probabilities are fixed).

If we now assume that $P_{in} = P$ is the vector of present state probabilities, then we should also have $P_{out} = P$, because the state line probabilities are constant in steady-state. If we assume that the state lines are *independent*, this translates to $P = F(P)$. The solution of this non-linear system gives the required state line probability vector P. It is solved using the Newton-Raphson method in [42], and using the Picard-Peano iteration method in [43].

Both techniques also try to correct for the state line independence assumption. In [42], this is done by accounting for m-wise correlations between state bits when computing their probabilities. This requires 2^m additional gates and can get very expensive. Nevertheless, they show good experimental results. The approach in [43] is to *unroll* the combinational logic block k times. This is less expensive than [42], and the authors observe that with $k = 3$ or so, good results can be obtained.

In order to avoid the problem of assuming that the FSM inputs are independent, the technique in [21] makes use of a user-specified input sequence. A new FSM is constructed that automatically generates the user input sequence, called an Input Modeling FSM (IMFSM). The combination of IMFSM and the original FSM are solved together as one autonomous FSM, using [42].

3.1.2. Statistical techniques

An alternative type of method was proposed in [27] that eliminates many of the shortcomings of the above probabilistic techniques. This is a statistical method in which the circuit is simulated repeatedly under randomly generated input vectors while monitoring the FF outputs, essentially a Monte Carlo approach. The simulation is stopped when the required FF probabilities have converged with user-specified accuracy and confidence.

The technique has many advantages: 1) it makes no assumptions about the FSM behavior (Markov or otherwise), 2) it makes no independence assumptions about the state lines, 3) it allows the user to specify the desired accuracy and confidence, and 4) it does not use large Binary Decision Diagrams [35] (BDDs), so that memory usage is not a problem.

For nodes inside the combinational block, only the steady state values (inside a clock cycle) are required. Therefore, a logic simulation using a zero-delay timing model may be safely used for the combinational block. In fact, the combinational block may be simulated at a higher level of abstraction, say as a single Boolean black box. The advantage of this is that the simulation can proceed much faster, which is important because the number of cycles to be simulated can be large. As a result, this approach has good speed and takes ≈4.5 hours to solve a ≈1500-latch/20k-gate circuit, on a SUN sparc-10, with 5% accuracy and 95% confidence. On small/moderate FSMs, the time required is in the seconds or minutes.

3.2. Combinational circuit power

Whereas flip-flop power is drawn in synchrony with the clock, the same is not true for gates inside the combinational logic. Even though the inputs to a combinational logic block are updated by the FFs (in synchrony with the clock), the internal gates of the block may make several transitions before settling to their steady state values for that clock period.

These additional transitions have been called hazards or glitches. Although unplanned for by the designer, they are not necessarily design errors. Only in the context of low-power design do they become a nuisance, because of the additional power that they dissipate. It has been observed [8] that this additional power dissipation is typically 20% of the total power, but can be as high as 70% of the total power in some cases such as combinational adders. We have observed that in a 16-bit parallel multiplier circuit, some nodes make as many as 20 transitions before reaching steady state. This component of the power dissipation is computationally expensive to estimate, because it depends on the timing relationships between signals inside the circuit. Consequently, many proposed power estimation techniques have ignored this issue. We will refer to this elusive component of power as the *glitch power*. Computing the glitch power is one main challenge in power estimation. This and other challenging problems that are specific to combinational circuit power estimation will be discussed below. In the second and third sub-sections, a survey of probabilistic and statistical techniques will be given.

3.2.1. Challenges

Recall the signal and transition probabilities, defined above, and suppose they are computed for every gate output node in the combinational block. It is important to note that the resulting values are unaffected by the circuit internal delays. This is because, by definition, they depend only on steady state signal values in a clock cycle. Indeed, these values would remain the same even if a *zero-delay timing model* were used. If this is done, however, the glitch power would be automatically excluded from the analysis. This is a serious shortcoming of techniques that are based on these measures, as we will point out below.

If a zero-delay model is assumed and the transition probabilities are computed, then the power can be computed as:

$$P_{av} = \frac{1}{2T_c} V_{dd}^2 \sum_{i=1}^{n} C_i P_t(x_i) \qquad (1)$$

where T_c is the clock period, C_i is the total capacitance at node x_i, and n is the total number of circuit nodes that are outputs of logic gates or cells. Since this assumes at most a single transition per clock cycle, then this is actually a *lower bound* on the true average power. Nevertheless, the results of a zero delay analysis may be useful as a rough technology-independent indication of the power requirements of a circuit, using estimated or nominal gate capacitances.

In order to compute the internal transition probabilities, it is common to start by finding the signal probabilities. This, by itself, is not easy and can be shown to be $\mathcal{NP}$-hard. The problem has to do with whether the input signals to a logic gate (viewed as *random variables*) are independent or not. In practice, logic signals may be correlated so that, for instance, two of them may never be simultaneously high, or they may never (or always) switch together. Primary inputs to the combinational block may be correlated due to the feedback. And even if these inputs are assumed independent, other internal signals may be correlated due to reconvergent fanout (a gate fans out into two signals that eventually recombine as the inputs of some gate downstream). However, it is computationally too expensive to compute these correlations.

Some have argued that the correlations do not seriously affect the final result, so that circuit input and internal nodes may be assumed to be *independent*. We refer to this as a *spatial independence* assumption. It leads to a significant simplification in computing the internal signal probabilities. If $y = ab$ is an AND gate output, and a and b are independent, then $P_s(y) = P_s(a)P_s(b)$. For an OR gate, we have $P_s(y) = P_s(a) + P_s(b)$. Thus the internal node probabilities are simply computed from those of the input nodes. The primary input node probabilities can be obtained as results of the analysis of the FSM, carried out previously.

To find the internal *transition* probabilities, we must deal with another independence issue of whether the values of the same signal in two consecutive clock cycles are independent or not. If assumed independent, then the transition probability can be easily obtained from the signal probability according to:

$$P_t(x) = 2P_s(x)P_s(\overline{x}) = 2P_s(x)\left[1 - P_s(x)\right] \qquad (2)$$

We refer to this as a *temporal independence* assumption. If this assumption is *not* made, then one must somehow represent the correlation between successive input vectors and internal signals. Given our formulation of power estimation as a two-step process, the correlation between two consecutive primary input bit values (on the same input line) can be obtained as transition probabilities computed during the FSM analysis. But that does not account for all input correlations. Correlations across more than one clock edge are not available, and correlations between one signal and previous values of other signals are also not available. In principle, the required correlation information is infinite, and only a finite amount of correlation can be considered in practice. Not only is computing the (limited) correlations too expensive, but making use of them during the computation of the combinational circuit power is also difficult.

The above problems become even worse in the case of non-zero delays. In this case, more detailed probability measures are required to properly formulate the power dissipation problem. One such measure is the *transition density* [25]. The transition density at node x is the average number of transitions per second at node x, denoted $D(x)$. Formally:

Definition 3. (transition density) *If a logic signal $x(t)$ makes $n_x(T)$ transitions in a time interval of length T, then the transition density of $x(t)$ is defined as:*

$$D(x) := \lim_{T \to \infty} \frac{n_x(T)}{T} \tag{3}$$

The density provides an effective measure of switching activity in logic circuits in the presence of any delay model. If the density at every circuit node is made available, the overall average power dissipation in the circuit can be computed as:

$$P_{av} = \frac{1}{2} V_{dd}^2 \sum_{i=1}^{n} C_i D(x_i) \tag{4}$$

In a synchronous circuit, with a clock period T_c, the relationship between transition density and transition probability is:

$$D(x) \geq \frac{P_t(x)}{T_c} \tag{5}$$

where equality occurs in the zero-delay case. Thus the transition probability gives a lower bound on the transition density.

In order to complete the density formulation, another measure is required: Let $P(x)$ denote the *equilibrium probability* [25] of a logic signal $x(t)$, defined as the *average fraction of time that the signal is high.* Formally:

Definition 4. (equilibrium probability) *If $x(t)$ is a logic signal (switching between 0 and 1), then its equilibrium probability is defined as:*

$$P(x) := \lim_{T \to \infty} \frac{1}{T} \int_{\frac{-T}{2}}^{\frac{+T}{2}} x(t)dt \tag{6}$$

In contrast to the signal probability, the equilibrium probability depends on the circuit internal delays since it describes the signal behavior over time, not only its steady state behavior per clock cycle. In the zero-delay case, the equilibrium probability reduces to the signal probability.

If all correlations are completely ignored, so that any two signals are completely independent both in space and time, we say that we have a *spatio-temporal independence assumption.* If this is assumed, then the transition density at the output y of a Boolean logic cell (gate) can be easily computed [25] from the density at its inputs, $x_1, \ldots, x_n$, according to:

$$D(y) = \sum_{i=1}^{n} P\left(\frac{\partial y}{\partial x_i}\right) D(x_i) \tag{7}$$

where $\partial y / \partial x$ is the *Boolean difference* of y with respect to x, defined as $\partial y / \partial x := y|_{x=1} \oplus y|_{x=0}$ where $\oplus$ denotes the exclusive-or operation.

3.2.2. Probabilistic techniques

Recently, several probabilistic power estimation techniques have been proposed for combinational circuits. These techniques all use simplified delay models for the circuit components and require user-supplied information about typical input behavior. Thus, their accuracy is limited by the quality of the delay models and of the input specification. Throughout the discussion below, primary inputs and primary outputs will refer to inputs and outputs of the combinational circuit block.

3.2.2.1. Using signal probability

In [19], a zero-delay model is used and temporal as well as spatial independence is assumed. The user is expected to provide signal probabilities at the primary inputs. These are then propagated into the circuit to provide the probabilities at every node. In the paper, the propagation of probabilities is performed at the switch-level, but this is not essential to the approach. It is easier to propagate probabilities by working with a gate-level description of the circuit. Once the signal probabilities are computed at every node in the circuit, the power is computed by making use of (1) and (2), based on the temporal independence assumption.

In general, if the circuit is built from Boolean components that are not part of a pre-defined gate library, the signal probability can be computed on the fly by using a BDD [35] to represent the Boolean functions, as proposed in [25] and [37]. Since it uses a zero-delay timing model, this method does not account for the glitch power.

3.2.2.2. Probabilistic simulation

A probabilistic power estimation approach that does compute the glitch power and does not make the zero-delay or temporal independence assumptions, called *probabilistic simulation* was proposed in [22]. This approach requires the user to specify typical signal behavior at the circuit inputs using *probability waveforms.* A probability waveform is a sequence of values indicating the probability that the signal is high for certain time intervals, and the probability that it makes low-to-high transitions at specific time points. The transition times themselves are not random. This allows the computation of the average, as well as the variance, of the current waveforms drawn by the individual gates in the design in one simulation run. The average current waveforms can then be used to compute the average power dissipated in each gate and the total average power of the circuit. Improvements on this technique were proposed in [23, 24], where the accuracy and the correlation handling were improved upon.

3.2.2.3. Transition density

In [25, 26], an efficient algorithm is presented to propagate the density values from the inputs throughout the circuit, according to (7). The required input specification is a pair of numbers for every input node, namely the equilibrium probability and transition density. In this case, both signal values and signal transition times are random.

BDDs can be used [25] to compute the Boolean difference probabilities, which are required in order to use the propagation algorithm (7). Recently, specialized BDD-based techniques have been proposed to facilitate this [39]. Improvements on this basic technique have also been proposed in [31, 51], providing more accurate gate models and improved correlation handling.

3.2.2.4. A symbolic technique

The technique proposed in [28] attempts to handle both spatial and temporal correlations by using a BDD to represent the successive Boolean functions at every node in terms of the primary inputs, as follows. The circuit topology defines a Boolean function corresponding to every node that gives the *steady state* value of that node in terms of the primary inputs. The intermediate values that the node takes before reaching steady state are not represented by this function. Nevertheless, one can construct Boolean functions for them by making use of the circuit delay information, assuming the delay of every gate is a specified fixed constant. As a result, the Boolean value at internal nodes is symbolically represented in terms of the primary inputs at all time points inside a clock cycle. In order to compute the probabilities of internal transitions, one can use the BDD [36] to construct the exclusive-OR function of two consecutive intermediate states.

One disadvantage of this technique is that it is computationally expensive. Since the BDD is built for the whole circuit, there will be cases where the technique breaks down because the required BDD may be too big.

3.2.2.5. Using correlation coefficients

Another probabilistic approach that is similar to probabilistic simulation was proposed in [24] whereby the correlation coefficients between steady state signal values (inside a clock cycle) are used as approximations to the correlation coefficients between the intermediate signal values (at any time during the clock cycle). This allows spatial correlation to be handled approximately, and is much more efficient than trying to estimate the dynamic correlations between intermediate states. The steady state correlations are estimated from the BDD by constructing the function for the AND of two signals. The reported results have good accuracy, but the technique does require building the BDD for the whole circuit, which may not always be feasible.

3.2.2.6 Handling spatio-temporal correlation

Finally, a probabilistic technique [52, 53] has been recently proposed that improves on the basic signal probability propagation method. This is done by using *transition probabilities* to account for temporal correlation (across one clock edge) and also using *correlation coefficients* in order to handle (approximately) the spatial and temporal correlation inside the circuit and improve the accuracy. The delay model remains zero delay, so that the glitch power is not included. Although this method uses BDDs, they only have to construct *local* BDDs in terms of the immediate fan-in inputs of every gate, so that there are no speed or memory problems.

3.2.3. Statistical techniques

The idea behind these techniques is quite simple and appealing: simulate the circuit repeatedly, using some timing or logic simulator, while monitoring the power being consumed. Eventually, the power will converge to the average power, based on (3) and (4). The issues are how to select the input patterns to be applied in the simulations and how to decide when the measured power has *converged* close enough to the true average power. Normally, the inputs are randomly generated and statistical mean estimation techniques [38] are used to decide when to stop - essentially a Monte Carlo method.

3.2.3.1. Total power

This approach [29, 30] uses Monte Carlo simulation to estimate the total average power of the circuit. It consists of applying randomly-generated input patterns at the primary inputs and monitoring the energy dissipated per clock cycle using a simulator, until the cumulative power measured has converged to the true average power. In practice, this technique was found to be very efficient. Typically, as few as 10 vectors may be enough to estimate the power of a large circuit with thousands of gates. But perhaps the most useful feature of this technique is that the user can specify the required accuracy and confidence level *up-front.* It also does not require an independence assumption for internal nodes. It only requires the primary inputs to be independent, but the approach can be extended to model and take into account the correlations between input nodes.

Perhaps the only disadvantage of this approach is that, while it provides an accurate estimate of the total power, it does not provide the power consumed by individual gates or small groups of gates. It would take many more transitions to estimate (with the same accuracy) the power of individual gates, because some gates may switch very infrequently.

3.2.2.2. Power of individual gates

This technique [32] is a modification of the above approach that provides both the total and individual-gate power estimates, with user-specified accuracy and confidence. One reason why one may want to estimate the power consumed by individual gates is to be able to *diagnose* a high power problem, and find out which part of the circuit consumes the most power. Other reasons have to do with the fact that estimating the individual gate power values is essentially equivalent to estimating the transition density values at all the nodes, which can then be used to estimate circuit reliability, considering a variety of failure mechanisms [25].

A weakness of this approach may be its moderate speed. For a circuit with 16000 gates, about 2 cpu hours are required on a SUN sparc ELC.

4. High-level power estimation

As pointed out in the introduction, it would be very advantageous if one could estimate power dissipation from a design description at a high level of abstraction. This would provide designers with an early measure of power dissipation, before much design effort has been spent. To date, some techniques have been proposed that work at the structural RTL level. At this level of abstraction, the memory elements (register files, flip-flops, etc) are assumed to have been completely specified, but all other (combinational) logic remains at the (Boolean) functional level. Essentially, the design description consists of flip-flops and Boolean black boxes.

In some cases, the high-level description of a design may be in terms of major circuit blocks that were used in previous designs. In this case, the detailed implementation of the combinational black boxes will be completely known. This is, for example, the case in DSP designs, where the circuit blocks come from a library of well characterized adders, multipliers, etc, and where the design task might be to determine which type of adder or multiplier to use in a given chip design. In this case, it is still advantageous to carry out the analysis at a high level of abstraction, because the analysis can be done much faster. I refer to techniques of this kind as being *bottom-up* approaches - the low-level details are *known*, but we choose to ignore them and use instead a simplified high-level model of the block behavior. This is essentially a *macro-modeling for power* approach. Bottom-up techniques have been proposed in [54, 55], where black-box models (macro-models) are built for circuit blocks by a process of characterization that models the block power as a function of the input/output signal statistics (probabilities) of the block. Other details are also included, such as the bus width, average capacitance, etc.

In other cases, the low-level details of the circuit blocks may be truly unknown yet, because such a circuit block my never have been designed before. This presents a harder problem to solve of extracting power out of pure (Boolean) functionality. I refer to such techniques as being *top-down*. Top-down techniques have been proposed in [44, 45], and make use of *entropy* of a logic signal as a measure of the amount of information that can be carried by that signal. The rationale for this is that the power requirements of a circuit must be related to the amount of *computational work* that the circuit performs, which has traditionally been modeled with the entropy measure. The entropy is directly related to the signal probability, so that the probabilities at the flip-flop outputs are used to compute the entropy input to the combinational blocks.

All these techniques are fairly recent, and it is not clear yet how useful they will be in practice, or how their performances compare/contrast in a practical setting.

5. Summary

Most proposed power estimation techniques operate at a low level of abstraction and use simplified delay models, so that they do not provide the same accuracy as, say, circuit simulation. But they are fast, which is very important because VLSI designers are interested in the power dissipation of large designs. Within the limitations of the simplified delay models, some of these techniques, e.g., the *statistical* techniques, can be very accurate. In fact the desired accuracy can be specified up-front. The other class of techniques, i.e., the *probabilistic* techniques, are not as accurate but can be faster.

From an implementation standpoint, one major difference between statistical and probabilistic techniques is that statistical techniques can be built around existing simulation tools and libraries, while probabilistic techniques cannot. Typically, probabilistic techniques require specialized simulation models.

Other, more recent, techniques have been proposed for high-level power estimation. Given a description of the design at the structural RTL level, these methods try to predict the power requirements that an implementation of this design would have.

References

[1] R. W. Brodersen, A. Chandrakasan, S. Sheng, "Technologies for personal communications," *Symp. on VLSI circuits*, pp. 5–9, 1991.

[2] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, April 1992.

[3] Workshop Working Group Reports, Semiconductor Industry Association, pp. 22–23, Nov. 1992.

[4] S. Chowdhury and J. S. Barkatullah, "Estimation of maximum currents in MOS IC logic circuits," *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 6, pp. 642–654, June 1990.

[5] S. Devadas, K. Keutzer, and J. White, "Estimation of power dissipation in CMOS combinational circuits using Boolean function manipulation," *IEEE Transactions on Computer-Aided Design*, vol. 11, no. 3, pp. 373–383, March 1992.

[6] H. Kriplani, F. N. Najm, and I. Hajj, "Pattern independent maximum current estimation in power and ground buses of CMOS VLSI circuits: algorithms, signal correlations, and their resolution," *IEEE Transactions on Computer-Aided Design*, vol. 14, no. 8, pp. 998–1012, August 1995.

[7] H. J. M. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 4, pp. 468–473, Aug. 1984.

[8] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of CMOS combinational logic networks," IEEE/ACM International Conference on Computer-Aided Design, pp. 402–407, November 1992.

[9] K. P. Parker and E. J. McCluskey, "Probabilistic treatment of general combinational networks," *IEEE Transactions on Computers*, vol. C-24, pp. 668–670, June 1975.

[10] S. Manne, A. Pardo, R. I. Bahar, G. D. Hachtel, F. Somenzi, E. Macii, and M. Poncino, "Computing the maximum power cycles of a sequential circuit," *32nd Design Automation Conference*, pp. 23–28, June 1995.

[11] S. M. Kang, "Accurate simulation of power dissipation in VLSI circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-21, no. 5, pp. 889–891, Oct. 1986.

[12] G. Y. Yacoub and W. H. Ku, "An accurate simulation technique for short-circuit power dissipation based on current component isolation," *IEEE International Symposium on Circuits and Systems*, pp. 1157–1161, 1989.

[13] A-C. Deng, Y-C. Shiau, and K-H. Loh, "Time domain current waveform simulation of CMOS circuits," *IEEE International Conference on Computer-Aided Design*, pp. 208–211, Nov. 1988.

[14] R. Tjarnstrom, "Power dissipation estimate by switch level simulation," *IEEE International Symposium on Circuits and Systems*, pp. 881–884, May 1989.

[15] U. Jagau, "SIMCURRENT - an efficient program for the estimation of the current flow of complex CMOS circuits," *IEEE International Conference on Computer-Aided Design*, pp. 396–399, Nov. 1990.

[16] T. H. Krodel, "PowerPlay - fast dynamic power estimation based on logic simulation," *IEEE International Conference on Computer Design*, pp. 96–100, October 1991.

[17] L. Benini, M. Favalli, P. Olivo, and B. Ricco, "A novel approach to cost-effective estimate of power dissipation in CMOS ICs," *European Design Automation Conference*, pp. 354–360, 1993.

[18] F. Dresig, Ph. Lanches, O. Rettig, and U. G. Baitinger, "Simulation and reduction of CMOS power dissipation at logic level," *European Design Automation Conference*, pp. 341–346, 1993.

[19] M. A. Cirit, "Estimating dynamic power consumption of CMOS circuits," *IEEE International Conference on Computer-Aided Design*, pp. 534-537, Nov. 1987.

[20] F. N. Najm and M. Y. Zhang, "Extreme delay sensitivity and the worst-case switching activity in VLSI circuits," *32nd Design Automation Conference*, pp. 623–627, June 1995.

[21] J. Monteiro and S. Devadas, "Techniques for the power estimation of sequential logic circuits under user-specified input sequences and programs," *ACM/IEEE International Symposium on Low Power Design*, pp. 33–38, April 1995.

[22] F. Najm, R. Burch, P. Yang, and I. Hajj, "Probabilistic simulation for reliability analysis of CMOS VLSI circuits," *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 4, pp. 439–450, April 1990 (Errata in July 1990).

[23] G. I. Stamoulis and I. N. Hajj, "Improved techniques for probabilistic simulation including signal correlation effects," *30th ACM/IEEE Design Automation Conference*, pp. 379–383, 1993.

[24] C-Y. Tsui, M. Pedram, A. M. Despain, "Efficient estimation of dynamic power consumption under a real delay model," *IEEE International Conference on Computer-Aided Design*, pp. 224–228, November 1993

[25] F. Najm, "Transition density: a new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design*, vol. 12, no. 2, pp. 310–323, February 1993.

[26] F. Najm, "Low-pass filter for computing the transition density in digital circuits," *IEEE Transactions on Computer-Aided Design*, vol. 13, no. 9, pp. 1123–1131, September 1994.

[27] F. N. Najm, S. Goel, and I. N. Hajj, "Power estimation in sequential circuits," *32nd Design Automation Conference*, pp. 635-640, June 1995.

[28] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," *29th ACM/IEEE Design Automation Conference*, pp. 253–259, June 1992.

[29] 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.

[30] R. Burch, F. Najm, P. Yang, and T. Trick, "A Monte Carlo approach for power estimation," *IEEE Transactions on VLSI Systems*, vol. 1, no. 1, pp. 63–71, March 1993.

[31] J-Y. Lin, T-C. Liu, and W-Z. Shen, "A cell-based power estimation in CMOS combinational circuits," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 304–309, November 1994.

[32] M. Xakellis and F. Najm, "Statistical Estimation of the Switching Activity in Digital Circuits," *31st ACM/IEEE Design Automation Conference*, pp. 728–733, 1994.

[33] S. M. Ross, *Stochastic Processes*, New York, NY: John Wiley & Sons, 1983.

[34] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 2nd Edition. New York, NY: McGraw-Hill Book Co., 1984.

[35] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Transactions on Computer-Aided Design*, pp. 677–691, August 1986.

[36] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient implementation of a BDD package," *27th ACM/IEEE Design Automation Conference*, pp. 40–45, June 1990.

[37] 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*, pp. 730–739, September 1989.

[38] I. Miller and J. Freund, *Probability and Statistics for Engineers*, 3rd edition. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1985.

[39] B. Kapoor, "Improving the accuracy of circuit activity measurement," *31st ACM/IEEE Design Automation Conference*, pp. 734–739, June 1994

[40] A. A. Ismaeel and M. A. Breuer, "The probability of error detection in sequential circuits using random test vectors," *Journal of Electronic Testing*, vol. 1, pp. 245–256, January 1991.

[41] G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Probabilistic analysis of large finite state machines," *31st ACM/IEEE Design Automation Conference*, pp. 270–275, June 1994.

[42] J. Monteiro and S. Devadas, "A methodology for efficient estimation of switching activity in sequential logic circuits," *ACM/IEEE 31st Design Automation Conference*, pp. 12–17, June 1994.

[43] C-Y Tsui, M. Pedram, and A. M. Despain, "Exact and approximate methods for calculating signal and transition probabilities in FSMs," *ACM/IEEE 31st Design Automation Conference*, pp. 18–23, June 1994.

[44] F. N. Najm, "Towards a high-level power estimation capability," *ACM/IEEE International Symposium on Low Power Design*, pp. 87–92, April 23-26, 1995.

[45] D. Marculescu, R. Marculescu, and M. Pedram, "Information theoretic measures of energy consumption at register transfer level," *ACM/IEEE International Symposium on Low Power Design*, pp. 81–86, April 1995.

[46] W. T. Eisenmann and H. E. Graeb, "Fast transient power and noise estimation for VLSI circuits," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 252–257, November 1994.

[47] H. K. Sarin and A. J. McNelly, "A power modeling and characterization method for logic simulation," *IEEE Custom Integrated Circuits Conference*, pp. 363–366, May 1995.

[48] A. Pardo, R. I. Bahar, S. Manne, P. Feldmann, G. D. Hachtel, and F. Somenzi, "CMOS dynamic power estimation based on collapsible current source transistor modeling," *ACM/IEEE International Symposium on Low Power Design*, pp. 111–116, April 1995.

[49] C. X. Huang, B. Zhang, An-Chang Deng, and B. Swirski, "The design and implementation of PowerMill," *ACM/IEEE International Symposium on Low Power Design*, pp. 105–109, April 1995.

[50] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: a first step towards software power minimization," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 384–390, November 1994.

[51] T-L. Chou, K. Roy, and S. Prasad, "Estimation of circuit activity considering signal correlations and simultaneous switching," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 300–303, November 1994.

[52] R. Marculescu, D. Marculescu, and M. Pedram, "Switching activity analysis considering spatiotemporal correlations," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 294–299, November 1994.

[53] R. Marculescu, D. Marculescu, and M. Pedram, "Efficient power estimation for highly correlated input streams," *32nd Design Automation Conference*, pp. 628–634, June 1995.

[54] P. E. Landman and J. M. Rabaey, "Activity-sensitive architectural power analysis for the control path," *ACM/IEEE International Symposium on Low Power Design*, pp. 93–98, April 1995.

[55] P. E. Landman and J. M. Rabaey, "Architectural power analysis: the dual bit type method," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 3, no. 2, pp. 173–187, June 1995.