Node Sampling: a Robust RTL Power Modeling Approach

Alessandro Bogliolo   Luca Benini

DEIS - Università di Bologna - Italy

Abstract

In this paper we propose a robust RTL power modeling methodology for functional units. Our models are consistently accurate over a wide range of input statistics, they are automatically constructed and can provide pattern-by-pattern power estimates. An additional desirable feature of our modeling methodology is the capability of accounting for the impact of technology variations, library changes and synthesis tools.

Our methodology is based on the concept of node sampling, as opposed to more traditional approaches based on input sampling. We analyze the theoretical properties of node sampling and we formally show that it is a statistically sound approach. The superior robustness of our method is due to its limited dependency on pattern-based characterization.

1 Introduction

Large digital circuits are nowadays often described and designed at the register-transfer level (RTL). RTL specifications can be partitioned in *instance-specific components*, such as controller state machines and sparse "glue logic" and *general-purpose functional macros*, such as adders, multipliers, FIR filter sections, etc. While instance-specific components are generally designed from scratch, general-purpose components can be re-used for many designs and are usually collected in *libraries*.

As designs get larger and time-to-market pressure increases, the fraction of general-purpose components tends to increase. In many cases, instance-specific blocks are replaced by programmable or re-configurable general-purpose components. This trend represents a good business opportunity for developers of general-purpose components that can be sold to a wide market of system designers. This new type of product is also known as *intellectual property* (IP) component. One of the key issues for the success of IP vendors, and more generally for the diffusion of an effective re-use paradigm, is the capability of providing information on critical cost metrics such as performance, area, testability and power.

When a designer instantiates an IP component, she/he wants to estimate what is the cost of the instantiation in term of the relevant cost metrics. In power-constrained systems, the power dissipation of each instance of each component should be estimated as accurately as possible. The most straightforward approach to power estimation for RTL components is to simulate their gate-level (or transistor-level) implementation. Unfortunately this solution has two main drawbacks. First, gate-level simulation is much slower and computationally more expensive than RTL simulation. Second, the IP provider needs to disclose to the user the full gate-level (transistor-level) netlist of the component. This may not be possible if the user is only evaluating the component to make a final decision on its purchase.

RTL power models are required to overcome these drawbacks. The fundamental requirements for practical RTL power models are the following. First, power estimation using RTL models should be much more efficient than gate-level (transistor-level) power estimation. Obviously, some accuracy loss can be tolerated. Second, the models should not require disclosure of the details of the internal structure of the IP component. Third, the models should be general and robust. In other words it should be possible for the IP provider to generate them with an automated procedure, and for the IP user to employ them with a wide variety of input pattern distributions without compromising accuracy.

Additional desirable features are *weak technology dependence* and *tunable accuracy*. Weak technology dependence is the property of rapidly obtaining a new set of models for an entire library of RTL components when the technology library used to map them is changed. Tunable accuracy is the capability of improving the accuracy of the power estimation at the RT level by increasing the amount of information provided and computational effort spent in model extraction.

We propose an approach to RTL power modeling that satisfies the fundamental requirements and provides in some degree the desirable features outlined above. Our approach is based on the key observation that power dissipation is strongly dependent on the *internal structure* of a component, and that *partial knowledge* of the internal structure is sufficient to provide accurate power estimations. Partial knowledge is obtained by *node sampling*. During model construction, the gate-level implementation of the component is examined, and a small subset of its internal nodes is selected (*i.e.*, a node sample). A model is constructed that estimates the total power consumption based on the switching activity of the node sample.

After model construction, power estimation is much more efficient than full gate-level power simulation, because only a partial and abstract representation of the internal functionality must be evaluated. IP protection is guaranteed because no information on the internal structure of the component is released (only a functional representation of a small number of internal nodes must be provided within the model). The model construction procedure is fully automatic. Most importantly, our RTL models are robust and accuracy is insensitive to the operating conditions. Additionally, our models are weakly technology-dependent and their accuracy can be finely tuned.

The paper is organized as follows. In the next section we briefly review previous power modeling approaches based on characterization. In Section 3 we give the theoretical foundations of node sampling. Several extensions to basic node sampling are described in Section 4 Experimental results are presented in Section 5. Finally, conclusions are drawn in Section 6.

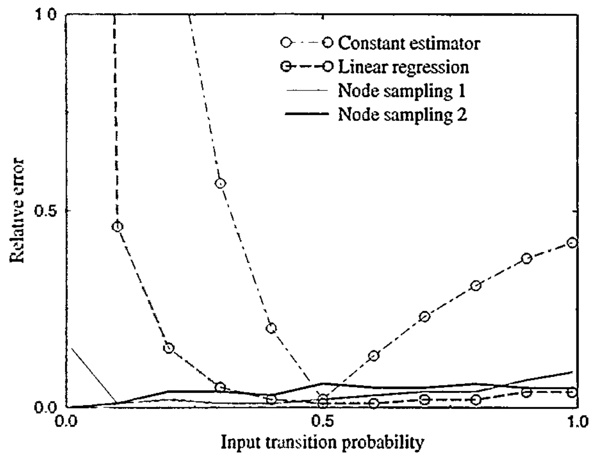

Figure 1: Relative error of different RTL power models of benchmark circuit cm85 as a function of its average input activity.

2 Related work

Several approaches to RTL power modeling have been proposed in the recent past. We will focus on those satisfying the fundamental requirements listed in the introduction [1, 2, 3, 4, 5, 6, 7]. In particular, the most challenging fundamental requirement is robustness. To increase robustness, previous approaches rely on some form of pattern-based *characterization*. In general terms, characterization is the basis for an interpolation process. During model construction, a gate-level (transistor-level) description of the component is simulated with a set of input patterns (*i.e.*, a *pattern sample*), and the power estimates obtained are exploited to obtain a model that can predict the power consumption for input patterns that are not in the sample.

Regression-based methods [6, 3] are motivated by the observed correlation between average input-output activity and average power. Power is modeled as a function of input-output activities and fitting parameters. The functional dependency on input-output switching may be linear or more complex [6, 3]. Characterization plays a fundamental role in regression-based approaches because it is used to adjust the value of the fitting parameters in the model. Regression-based power models are highly accurate if evaluated in the same operating conditions used for characterization, while their accuracy decreases when the input statistics change.

This problem is best clarified through a simple example. Figure 1 illustrates the limited robustness of regression-based power models. The relative error on power estimates is plotted as a function of the average input activity. The two dashed curves refer to two RTL power models (namely, the *constant model* [1] and the *linear regression model* [6, 3]) characterized for MCNC91 [8] benchmark cm85. Both models were characterized on a sample of power consumption data obtained by simulating the gate-level implementation for a set of random input patterns with 0.5 average transition probability. In the constant model, the power values of the sample were averaged and the average value was taken as a (constant, pattern independent) power estimator. The linear regression model attempts to take into account the input pattern dependence by modeling power as a linear function of input activity[1].

Gate-level and RTL simulation were repeatedly performed (with different input sequences representing different operating conditions) to evaluate the accuracy of the power estimates provided by the models. Each point in Figure 1 represents the result of a simulation run performed by applying an input sequence of 10000 vectors with the average transition probability reported on the abscissa. The relative error is small (around 2%) for input activities similar to those used for characterization (*i.e.*, 0.5), but it rapidly increases as the statistics of the input patterns change, becoming much larger than 100% for low input activities. The linear regression model is more robust than the constant model, but it still suffers from characterization-induced limitations. The root mean square of the relative errors provided by a power model on a wide range of operating conditions is a meaningful measure of the model robustness. We call this measure *root mean square relative error*, denoted by RMSRE. For the constant (linear) model of Figure 1 the RMSRE is of 1600% (281%).

[1]In symbols $P(X^i, X^f) = \alpha_0 + \alpha_1(x_1^i \oplus x_1^f) + \alpha_2(x_2^i \oplus x_2^f) + \cdots + \alpha_n(x_n^i \oplus x_n^f)$ Coefficients $\alpha_0, \alpha_1, \cdots, \alpha_n$ are computed by least mean square fitting on the power sample [6]

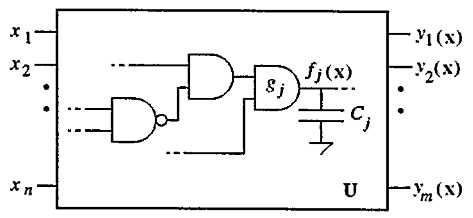

Figure 2: Model of a generic combinational unit.

More robust advanced characterization-based methods have been formulated in the recent past. The *lookup-table method* [5] increases robustness by running multiple characterization experiments for a large set of different input statistics, and by storing multiple power models in a lookup table addressed by a compact signature (namely, average input probability, average input switching activity and average output switching activity). Although this model is robust and accurate, signature extraction imposes a loss of information on the distribution of input patterns, which may cause accuracy loss for highly biased pattern sets. Moreover, it is not clear how accuracy is traded off for lookup table size. Another approach to improve robustness is to exploit a multi-level simulation engine to perform *in-situ tuning* of pre-characterized models [4, 7]. Although this approach appears to be very effective, it requires a multi-level simulator which may not be available and imposes a non-negligible run-time overhead.

To improve model robustness (*i.e.*, to obtain lower RMSRE), we move from the observation that power dissipation for a digital unit can be inferred by monitoring the activity of a few internal nodes. Hence, we adopt a new viewpoint. We do not sample on input patterns, but *we sample on internal nodes*. The improved robustness obtained by *node sampling* is shown in Fig. 1: curves "Node sampling 1" and "Node sampling 2" refer to energy estimates obtained by monitoring the activity at 4 and 8 internal nodes, respectively. In the next section we show that this intuitively appealing viewpoint has sound theoretical foundations and we prove some desirable statistical properties of our method.

3 Node sampling

3.1 Theoretical basis

Consider a combinational CMOS unit U with input vector $\mathbf{x} = (x_1, ...x_n)$ for which an input-output behavioral model is provided, as well as its gate-level implementation, as shown in Figure 2. We assume that the unit is stable at time t^i and t^f ($t^f > t^i$) and that an input transition from $\mathbf{x}^i$ to $\mathbf{x}^f$ occurs between t^i and t^f. We denote by $\epsilon(\mathbf{x}^i, \mathbf{x}^f)$ the supply energy drawn by the gate-level implementation in the time interval $[t^i, t^f]$. The task of modeling power consumption at the RT level consists of finding a simple but accurate model for $\epsilon(\mathbf{x}^i, \mathbf{x}^f)$ (the corresponding power consumption being $p(\mathbf{x}^i, \mathbf{x}^f, T) = \epsilon(\mathbf{x}^i, \mathbf{x}^f)/T$, where $T = t^f - t^i$). In the following treatment, we assume $T = 1$ and $V_{dd} = 1$, thus energy and power have the same numeric value, and the constant V_{dd}^2 disappears from the computations. Different values of T and V_{dd} change the formulas only by constant multiplicative factors.

If we consider the internal structure of the unit, $\epsilon(\mathbf{x}^i, \mathbf{x}^f)$ can be expressed as

$$\epsilon(\mathbf{x}^i, \mathbf{x}^f) = \sum_{j=1}^{N} \epsilon_j(\mathbf{x}^i, \mathbf{x}^f) \quad (1)$$

where $\epsilon_j(\mathbf{x}^i, \mathbf{x}^f)$ is the energy drawn by gate g_j as a function of the input transition, and N is the number of gates in the unit. We focus, at first, on a given input transition $(\bar{\mathbf{x}}^i, \bar{\mathbf{x}}^f)$, thus temporarily removing pattern dependence from our notation (ϵ and ϵ_j will be used to denote $\epsilon(\bar{\mathbf{x}}^i, \bar{\mathbf{x}}^f)$ and $\epsilon_j(\bar{\mathbf{x}}^i, \bar{\mathbf{x}}^f)$, respectively).

In general, the ϵ_j's are not available at the RTL and Equation (1) is not an RTL model for ϵ. We observe, however, that a simple algebraic manipulation allows us to represent ϵ as the mean of the finite set $\mathcal{E} = \{N \cdot \epsilon_1, N \cdot \epsilon_2, ..., N \cdot \epsilon_N\}$, hereafter called *energy population* associated with input transition $(\bar{\mathbf{x}}^i, \bar{\mathbf{x}}^f)$:

$$\epsilon = \sum_{j=1}^{N} \epsilon_j = \frac{1}{N} \sum_{j=1}^{N} N \cdot \epsilon_j \quad (2)$$

If statistical information is available about the distribution of $\mathcal{E}$, statistical inference can be used to estimate its mean value ϵ, thus solving the energy estimation problem. This is the basic idea behind *node sampling.*

Definition 1 *We call* **node-sampling experiment** *the uniform random selection of a node (say g_j) of the given unit. We call* **energy-sampling experiment** *(induced by node-sampling) the selection of the element of $\mathcal{E}$ associated with the outcome of a node-sampling experiment.*

The outcome of an energy-sampling experiment is a value of a discrete random variable R, defined on energy population $\mathcal{E}$, The *probability distribution* of R is a function $F(r) = prob\{R \leq r\}$ that represents the fraction of nodes with energy consumption $\epsilon_j \leq r/N$. The corresponding *probability function* $P(r) = prob\{R = r\}$ represents the fraction of nodes with energy consumption $\epsilon_j = r/N$.

Theorem 1 *The total energy ϵ is the expected value of random variable $\mathcal{R}$, i.e., the mean of the energy population.*
Proof: *the formal proof directly comes from the definition of expected value ($E[R]$) and from Equation (2)*

$$E[R] = \sum_{r \in \mathcal{E}} r \cdot P(r) = \sum_{j=1}^{N} N \cdot \epsilon_j \frac{1}{N} = \sum_{j=1}^{N} \epsilon_j = \epsilon$$

Definition 2 *We call* **random energy sample** *$\mathcal{S}_{\mathcal{E}}$ of size s from the energy population $\mathcal{E}$ the set of random variables $R_1, R_2, ..., R_s$ associated with s independent energy-sampling experiments.*

Any function Θ of a random sample used to estimate an unknown parameter θ of the parent population is said to be an *estimator* for θ. Our target is that of finding an *unbiased, efficient* estimator of ϵ [9].

An estimator is said to be unbiased if and only if, on average, it is on target. In symbols, Θ is an unbiased estimator for ϵ if and only if

$$E[\Theta(R_1, R_2, ..., R_s)] = \epsilon$$

Efficiency is a measure of the quality of the estimator. Any observed value of the estimator ($\bar{\Theta}$) is a *point estimate* of the target parameter ϵ. If the estimator is efficient, the point estimates it provides have a high probability of being close to the target.

In the rest of this section we demonstrate that the mean of a random energy sample provides unbiased and efficient estimates for ϵ. The problem of collecting random energy samples at the RTL will be addressed in Subsection 3.2.

Theorem 2 *The sample mean $\overline{R}$ of $\mathcal{S}$ is an unbiased estimator for ϵ.*
Proof: *By Theorem 1, we already know that ϵ is the population mean $E[R]$. Here we need only to prove that $\overline{R}$ is an unbiased estimator for $E[R]$, i.e., that $E[\overline{R}] = E[R]$:*

$$E[\overline{R}] = E\left[\frac{\sum_{i=1}^{s} R_i}{s}\right] = \frac{1}{s}\sum_{i=1}^{s} E[R_i]$$
$$= \frac{1}{s}\sum_{i=1}^{s} E[R] = E[R]$$

Hence, $E[\overline{R}] = E[R] = \epsilon$.

Efficiency can be expressed in terms of *mean square error* (MSE). By definition, the MSE of an unbiased estimator is equal to its variance. Given two estimators Θ_1 and Θ_2 of the same parameter, Θ_1 is said to be more efficient than Θ_2 if $Var[\Theta_1] < Var[\Theta_2]$.

The variance of the sample mean is equal to the population variance divided by the sample size [9]. If we denote by σ^2 the variance of the energy population, the variance (*i.e.*, the MSE) of $\overline{R}$ is σ^2/s. In practice, node sampling provides a family of unbiased estimators for ϵ, whose efficiency grows with the sample size. The minimum sample size reuired to achieve a target accuracy (MSE = ϵ^2) is

$$s \geq \frac{\sigma^2}{\epsilon^2} \quad (3)$$

We remark that the target of the estimators described so far is the energy consumption of the unit corresponding to a given input transition. We assume that such estimators have been constructed for all possible input transitions and we use $\Theta(\mathbf{x}^i, \mathbf{x}^f)$ to denote the estimator of $\epsilon(\mathbf{x}^i, \mathbf{x}^f)$.

$\Theta(\mathbf{x}^i, \mathbf{x}^f)$ is a *pattern-dependent* model for ϵ, whose point accuracy depends on the individual efficiency of each estimator. The overall accuracy provided by the estimator when evaluating the average energy drawn by the circuit in a long period of time is usually higher.

If a test sequence $\mathcal{T}$ of $m+1$ input patterns (m input transitions) is applyed to the unit, the average energy estimate is the average of the pattern-by-pattern point estimates provided by the estimators corresponding to each transition. In symbols, an estimator $\Theta_{avg}(\mathcal{T})$ for $\epsilon_{avg}(\mathcal{T})$ can be defined as follows:

$$\Theta_{avg}(\mathcal{T}) = \frac{1}{m} \sum_{(\mathbf{x}^i, \mathbf{x}^f) \in \mathcal{T}} \Theta(\mathbf{x}^i, \mathbf{x}^f) \quad (4)$$

Theorem 3 *$\Theta_{avg}(\mathcal{T})$ is an unbiased estimator for $\epsilon_{avg}(\mathcal{T})$.*
Proof: *By linearity, the expected value of $\Theta_{avg}(\mathcal{T})$ can be expressed in terms of the expected values of the pattern-by-pattern estimators:*

$$\begin{aligned} E[\Theta_{avg}(\mathcal{T})] &= E\left[\frac{1}{m} \sum_{(\mathbf{x}^i, \mathbf{x}^f) \in \mathcal{T}} \Theta(\mathbf{x}^i, \mathbf{x}^f)\right] \\ &= \frac{1}{m} \sum_{(\mathbf{x}^i, \mathbf{x}^f) \in \mathcal{T}} E[\Theta(\mathbf{x}^i, \mathbf{x}^f)] \end{aligned}$$

On the other hand we know by Theorem 2 that any $\Theta(\mathbf{x}^i, \mathbf{x}^f)$ is an unbiased estimator for the corresponding energy. Hence,

$$\begin{aligned} \frac{1}{m} \sum_{(\mathbf{x}^i, \mathbf{x}^f) \in \mathcal{T}} E[\Theta(\mathbf{x}^i, \mathbf{x}^f)] &= \frac{1}{m} \sum_{(\mathbf{x}^i, \mathbf{x}^f) \in \mathcal{T}} \epsilon(\mathbf{x}^i, \mathbf{x}^f) \\ &= \epsilon_{avg}(\mathcal{T}) \end{aligned}$$

Notice that we didn't make any assumption about the input sequence $\mathcal{T}$ to demonstrate Theorem 3. Hence we can state that *the estimates provided by $\Theta_{avg}(\mathcal{T})$ are always unbiased, independently of the input statistics.* This is the main distinguishing feature of the energy estimator we propose in this paper.

3.2 Model construction

The estimator described so far is not yet an RTL model. In this section we address the practical issues of constructing an RTL energy model that retains the statistical properties of $\Theta_{avg}(\mathcal{T})$.

There are three main issues involved in constructing an RTL energy model of practical interest from $\Theta_{avg}(\mathcal{T})$:

1. how to perform the *energy-sampling* experiments,
2. how to construct and represent energy estimators for all possible input transitions,
3. how to determine the size of the random energy samples required to achieve the desired efficiency.

First of all, we need a reference model for the energy drawn by a generic gate g_j. The model we propose is

$$\epsilon_j(\mathbf{x}^i,\mathbf{x}^f) = C_j f_j(\mathbf{x}^i) \oplus f_j(\mathbf{x}^f) \tag{5}$$

where $f_j(\mathbf{x})$ is the function realized at gate g_j and C_j is the load capacitance of the gate (see Fig. 2). In the general case, when $V_{dd} \neq 1$, the load capacitance is multiplied by V_{dd}^2. Operator $\oplus$ is the Boolean exclusive-or. Equation 5 models the *zero-delay contribution* to the energy dissipated by gate g_j, and it does not account for glitch power and other parasitic phenomena. Assuming zero-delay power model implies that the RTL model can only predict the zero-delay dynamic power consumption of the gate-level implementation of unit U. In many cases this is sufficient for the accuracy required at the RT level. In Section 4 we will extend the model to deal with full-delay models and parasitic phenomena.

Given the energy model for all nodes, energy sampling consists of randomly choosing a set of gates and evaluating the corresponding *scaled energy contributions* $N \cdot \epsilon_j(\mathbf{x}^i,\mathbf{x}^f)$. In principle, this procedure has to be repeated for any input transition applied to the unit during RTL simulation. In practice, however, performing random sampling at the RTL is impractical because it implies the knowledge of the Boolean functions realized at all internal nodes. To overcome this drawback, we propose to perform node sampling only once for all, during model construction.

Definition 3 *We call* **node sample** $\tilde{\mathcal{S}}_N$ *of size s a set of s gates of the unit randomly selected by means of independent node-sampling experiments.*

Observation 1 *For any input transition $(\mathbf{x}^i,\mathbf{x}^f)$, the set of scaled energy values associated with node sample $\tilde{\mathcal{S}}_N$ is an observation $\tilde{\mathcal{S}}_\mathcal{E}$ of a random energy sample $\mathcal{S}_\mathcal{E}$ of size s from the energy population $\mathcal{E}(\mathbf{x}^i,\mathbf{x}^f)$. The sample mean of $\tilde{\mathcal{S}}_\mathcal{E}$ is an observation of $\Theta(\mathbf{x}^i,\mathbf{x}^f)$, i.e., an unbiased point estimate of $\epsilon(\mathbf{x}^i,\mathbf{x}^f)$.*

$$\begin{aligned}\tilde{\Theta}(\mathbf{x}^i,\mathbf{x}^f) &= \frac{1}{s}\sum_{g_j\in\tilde{\mathcal{S}}_N} N\cdot\epsilon_j(\mathbf{x}^i,\mathbf{x}^f) \\ &= \frac{1}{s}\sum_{g_j\in\tilde{\mathcal{S}}_N} N\cdot C_j f_j(\mathbf{x}^i)\oplus f_j(\mathbf{x}^f)\end{aligned} \tag{6}$$

Equation (6) holds for all input transitions. This is the key property that makes node sampling an RTL model of practical interest: once a node sample has been selected, the knowledge of the logic functions realized at nodes in the sample (that are a small fraction of the internal gates of the unit) provides sufficient information to obtain unbiased point estimates for the energy consumption of the unit corresponding to any input transition. Hence, Equation (6) is the RTL energy model we were looking for. Its evaluation requires the computation of s Boolean functions, a much easier task than computing the gate-level power consumption for the entire unit. Moreover, if the gate-level implementation is a third party intellectual property, IP protection is ensured because information on the gate-level structure is completely lost. Finally, the model is fully analytic and does not require any form of characterization and fitting on simulation results.[2]

In practice, Equation (6) says that the energy drawn by the entire unit is (on average) proportional to the *sample energy* $e_S(\mathbf{x}^i,\mathbf{x}^f)$, that is the energy drawn by gates in the sample:

$$\frac{1}{s}\sum_{g_j\in\tilde{\mathcal{S}}_N} N\cdot\epsilon_j(\mathbf{x}^i,\mathbf{x}^f) = \frac{N}{s}e_S(\mathbf{x}^i,\mathbf{x}^f)$$

Using a static node sample has several consequences that need to be discussed. First, Equation (6) represents point estimates rather than estimators: whenever the same input transition is applyed to the circuit the same energy estimate is returned by the model. This observation does not impair the desirable statistical properties of the model: point estimates are unbiased and their efficiency can be chosen by tuning the sample size.

Second, the same sample size is used for all input transitions. According to Equation (3), the minimum sample size required to achieve the desired efficiency depends on the population variance σ^2. Since the energy population (and its variance) depends on the input transition, the sample size has to be decided based on the maximum variance over all possible transitions

$$\sigma^2_{max} = \max_{(\mathbf{x}^i,\mathbf{x}^f)}\{\sigma^2(\mathbf{x}^i,\mathbf{x}^f)\}$$

The exact computation of σ^2_{max} is not an easy task. In principle it requires either an exhaustive gate-level simulation, or the solution of satisfiability problems. In the next subsection we construct an upper bound ($\bar{\sigma}^2_{max}$) for $\sigma^2(\mathbf{x}^i,\mathbf{x}^f)$ that can be statically computed to obtain a conservative estimate of σ^2_{max}. The sample size sufficient to obtain pattern-dependent energy estimates with MSE of ϵ^2 will be:

$$s \geq \frac{\bar{\sigma}^2_{max}}{\epsilon^2} \tag{7}$$

3.3 Computing an upper bound for σ^2

Referring to Equation (5), the energy population associated with the generic input transition is:

$$\mathcal{E}(\mathbf{x}^i,\mathbf{x}^f) = \{N\cdot C_j\cdot f_j(\mathbf{x}^i)\oplus f_j(\mathbf{x}^f) | 1\leq j\leq N\}$$

where the exclusive OR's $f_j(\mathbf{x}^i)\oplus f_j(\mathbf{x}^f)$ can take value 0 or 1.

We call *generalized energy population* (g.e.p.) a parameterized set of scaled energy values

$$\mathcal{E}_g(\delta_1,\delta_2,...,\delta_N) = \{N\cdot C_1\cdot\delta_1, N\cdot C_2\cdot\delta_2,...,N\cdot C_N\cdot\delta_N\}$$

where parameters δ_j represent Boolean flags. We denote by $\mathbf{d}$ the vector of the N Boolean flags, and by $\bar{\sigma}^2(\mathbf{d})$ the variance of the g.e.p. The maximum of $\bar{\sigma}^2(\mathbf{d})$ is an upper bound for σ^2.

[2] Characterization will be introduced in Section 4 to model glitch power and parasitic phenomena.

Theorem 4 *The maximum variance of $\mathcal{E}_g(\mathbf{d})$ (over all configurations of* $\mathbf{d}$*) is an upper bound for the maximum variance of $\mathcal{E}(\mathbf{x}^i, \mathbf{x}^f)$ (over all configurations of $(\mathbf{x}^i, \mathbf{x}^f)$):*

$$\max_{(\mathbf{x}^i,\mathbf{x}^f)} \{\sigma^2(\mathbf{x}^i, \mathbf{x}^f)\} \leq \max_{\mathbf{d}}\{\tilde{\sigma}^2(\mathbf{d})\}$$

Proof: *To prove the theorem it is sufficient to show that for each configuration of $(\mathbf{x}^i, \mathbf{x}^f)$ there exists a configuration of* $\mathbf{d}$ *such that $\mathcal{E}(\mathbf{x}^i, \mathbf{x}^f) = \mathcal{E}_g(\mathbf{d})$. Such a configuration can always be obtained by setting $\delta_j = f_j(\mathbf{x}^i) \oplus f_j(\mathbf{x}^f)$ for all j's. If $(\bar{\mathbf{x}}^i, \bar{\mathbf{x}}^f)$ is the input transition that maximizes the variance of $\mathcal{E}(\mathbf{x}^i, \mathbf{x}^f)$, a configuration $\bar{\mathbf{d}}$ exists such that $\mathcal{E}_g(\bar{\mathbf{d}})$ has the same variance. This actually demonstrate that the maximum variance of $\mathcal{E}_g$ is greater or equal than the maximum variance of $\mathcal{E}$.*

Finding the maximum variance of $\mathcal{E}_g(\mathbf{d})$ is a much easier task than finding the maximum variance of $\mathcal{E}(\mathbf{x}^i, \mathbf{x}^f)$, because it does not depend on gates' functionalities. Moreover, we can restrict our search to a small subset of g.e.p.'s that we call *unbalanced energy populations* (u.e.p.'s).

A g.e.p. is said to be unbalanced if and only if

$$\delta_j = 1, \delta_h = 0 \Rightarrow N \cdot C_h \leq N \cdot C_j$$

In other words, an u.e.p. is a g.e.p. where non-zero flags are associated with the higher values of $N \cdot C_j$.

Lemma 1 *The g.e.p. of maximum variance is an u.e.p.*

For the sake of conciseness we do not report the formal proof of the Lemma, but we observe that u.e.p.'s are intuitively more spread than other g.e.p.'s.

For a given unit with N nodes, there are only $N+1$ u.e.p.'s. If node capacitances are sorted in decreasing order ($j < h \Rightarrow C_j \geq C_h$), we can use an integer index k to uniquely identify u.e.p.'s: $\mathcal{E}_u(k)$ is an unbalanced energy population with $\delta_j = 1$ iff $j \leq k$.

All u.e.p.'s can be visited in linear time while iteratively computing their variance. In we denote by σ_k^2 and μ_k the variance and mean of $\mathcal{E}_u(k)$, respectively, the following relations hold:

$$\begin{cases} \mu_0 = 0 \\ \mu_k = \mu_{k-1} + C_k \end{cases}$$

$$\begin{cases} \sigma_0^2 = 0 \\ \sigma_k^2 = \sigma_{k-1}^2 + \frac{(N \cdot C_k)^2}{N} - C_k^2 - 2\mu_{k-1}C_k \end{cases} \tag{8}$$

Lemma 1 and Equation (8) allow us to determine in linear time an upper bound for the variance of the energy population, based only on static information (namely, the load capacitances of the gates in the unit):

$$\tilde{\sigma}_{max}^2 = \max_{0 \leq k \leq N} \{\sigma_k^2\}$$

In our experiments the upper bound was, on average, twice the maximum value of σ^2, leading to conservative sample sizes.

4 Extensions

In this section we propose two significative extensions to the basic node sampling approach. The extensions improve both accuracy and flexibility of our method. On the other hand, however, they rely on pattern-based characterization. As a consequence, the rigorous results obtained in the previous section lose validity, and accuracy is not theoretically bound any more.

4.1 Delay-sensitive second-order contributions

In the previous section we have described an RTL power model of the zero-delay gate-level power consumption. In other words, $N/s \cdot e_S(\mathbf{x}^i, \mathbf{x}^f)$ is a good predictor of the power dissipation estimated by zero-delay simulation of the golden model. It is a well-known fact that zero-delay power estimation may be inaccurate because of glitch power consumption and short-circuit currents. Such contributions need to be modeled as well to increase the accuracy of the RTL model. We move from the observation that second-order phenomena can be seen as an additive contribution (called ϵ^1) to zero-delay energy e:

$$e^{fd}(\mathbf{x}^i, \mathbf{x}^f) = e(\mathbf{x}^i, \mathbf{x}^f) + e^1(\mathbf{x}^i, \mathbf{x}^f)$$

where e^{fd} denotes the *full-delay* energy. The straight-forward extension of our estimates to full-delay models is:

$$\tilde{\Theta}^{fd}(\mathbf{x}^i, \mathbf{x}^f) = \frac{N}{s}(e_S(\mathbf{x}^i, \mathbf{x}^f) + e_S^1(\mathbf{x}^i, \mathbf{x}^f))$$

The RTL model we propose for e_S^1 is a regression equation that needs to be characterized on the results of gate-level simulations. In practice, we propose a hybrid modeling approach for the energy drawn by the sample, where simulation-based characterization of e_S^1 improves the absolute accuracy of the fully-analytic characterization-free model of e_S based on node sampling.

Experimentally, we observed that e_S^1 correlates well with input switching activity, hence we adopt the following model:

$$e_S^1(\mathbf{x}^i, \mathbf{x}^f) = \alpha \cdot d_H(\mathbf{x}^i, \mathbf{x}^f) \tag{9}$$

where $d_H(\mathbf{x}^i, \mathbf{x}^f)$ is the Hamming distance between $\mathbf{x}^i$ and $\mathbf{x}^f$ (*i.e.*, the number of switching inputs) and α is a fitting coefficient that is computed with a simple characterization experiment. The unit is simulated with a sample of random patterns. Full-delay simulation is employed. For each pattern pair $(\mathbf{x}^i, \mathbf{x}^f)$, an estimate of e_S^1 is computed by subtracting the zero delay energy (provided by the zero-delay model e_S) from the total energy computed by the simulator. An estimate for α is obtained by computing the ratio e_S^1/d_H for each pattern pair in the sample, then averaging the ratios. More formally, α is computed by linear least-squares fitting on the residual errors of the zero-delay model.

4.2 Technology tuning

One of the assumptions made in the model construction process described so far is the availability of a *golden model* (*i.e.*, a reference gate-level implementation of U). A *golden model* is always available because the IP provider that builds the model has complete access on the implementative details of the unit. However, this assumption implies that the RTL power model is a good predictor of the power dissipation of the *golden model* used for its construction. This may be an undesirable situation, because IP vendors do not always provide components mapped and bound to a particular technology and gate library. On the contrary, IP components are often designed and sold as *soft macros*, *i.e.*, abstract RTL descriptions (usually specified using a hardware description language such as Verilog HDL or VHDL) that can be synthesized by the user using a proprietary technology and gate library.

In the case of soft macros, there is no guarantee that the *golden model* employed for model construction is the same that will be instantiated by the user when he/she synthesizes the soft macro. There are three main sources of mismatch: the technology may be different (for instance, more aggressively scaled), the library containing the gate-level primitives used for synthesis may differ in richness and type of primitives and the synthesis tools employed to obtain the final implementation may be different or operated with different constraints and settings. As a result, the power estimates provided by the RTL model may be inaccurate if no corrective measure is taken.

Fortunately, accuracy may be recovered thanks to a *technology tuning* method that requires only limited effort on the IP user side. Our approach is based on the observation that the mismatches in final implementations produced by technology, library and synthesis tools tend to have limited variance, although their absolute value can be significant. Technology tuning is performed using a reference benchmark $\mathcal{B}$, a macro that contains no intellectual property and that can be released by the IP provider to the IP user at no charge. A set of input patterns and the corresponding average power dissipation P_{prov} will be provided with $\mathcal{B}$. The power dissipation is computed by the IP provider performing gate-level simulation of an implementation of $\mathcal{B}$. The implementation of $\mathcal{B}$ is obtained using the same technology, library and synthesis tools that are employed for generating the *golden models* of the IP units.

To perform technology tuning, the IP user synthesizes $\mathcal{B}$ with his/her technology, library and synthesis tools, then simulates the implementation with the patterns provided and measures the average power dissipation P_{user}. The *technology scaling parameter* S_{tech} is defined as $S_{tech} = P_{user}/P_{prov}$. Technology tuning is performed once for all. S_{tech} can be used for an entire library of IP macros. No intellectual property is disclosed to the user and the user can perform technology tuning before purchasing the IP macros. After S_{tech} has been computed, the user can estimate the power consumption of his/her own implementation of the IP soft macros with the following formula:

$$\epsilon_{mod}^{user}(x^i, x^f) = S_{tech} \cdot \epsilon_{mod}^{prov}(x^i, x^f) \qquad (10)$$

where ϵ_{mod}^{prov} is the power dissipation estimate of the model provided by the IP provider with the unit (Equation 6).

5 Experimental results

The modeling approach and algorithms presented in this paper have been implemented and tested on all MCNC91 [8] benchmark circuits. Each benchmark was first mapped on a reference technology library to obtain a *golden model*. Zero-delay power models were then automatically constructed by node sampling. The number of nodes to be included in the sample is decided a-priori based on the capacitance distribution for the nodes in the golden model, as explained in Section 3. In order to achieve comparable relative accuracy for all benchmarks, we used a target MSE parameterized on a static estimate of the average energy consumption of the unit. We used the mean ($\bar{\mu}$) of the maximum-variance g.e.p. of Section 3.3, as an estimate of the average energy consumption. The accuracy specification was

$$MSE = \epsilon^2 = (0.5 \cdot \bar{\mu})^2$$

from which a sample size was obtained for each benchmark according to Equation (3):

$$s = 4\frac{\bar{\sigma}_{max}^2}{\bar{\mu}^2}$$

In practice, the parameterized MSE specifies a target relative accuracy of 50% for the pattern-by-pattern energy estimates. We also imposed a hard upper bound on the number of nodes in the sample: maximum 10% of the nodes in the circuit were selected. This is because the bound of Equation (3) can be overly conservative due to the fact that it is based on the error on pattern-by-pattern power estimation, while target of our experiments was accurate average power estimation. The averaging over multiple patterns further reduces the variance and tightens the error bound.

Model construction consists of randomly selecting internal nodes. The run time for this process is negligible. Constant and linear estimators were also characterized for comparisons. Characterization was performed by fitting power data provided by gate-level simulations of the *golden models* with random input sequences with 0.5 average activity.

Circuit		RMS Rel. Err. (%)			
Name	Size	Const	Lin	Ns	Ns$^{\mathcal{B}}$
`cm85`	31	1600.3	281.5	**6.1**	**1.6**
`cm150`	46	1303.1	259.3	**6.9**	**9.6**
`cmb`	34	1533.1	121.6	**3.9**	**4.1**
`parity`	36	735.4	267.9	**12.7**	**11.2**
`mux`	61	1194.0	171.6	**3.0**	**4.8**
`alu2`	252	1108.2	381.0	**8.8**	**5.5**
`alu4`	460	939.5	361.7	**6.4**	**8.5**
`c432`	159	889.7	278.8	**5.2**	**4.3**
`c880`	211	877.8	254.1	**6.0**	**4.0**
`c1355`	521	363.8	180.7	**4.6**	**4.7**
`c7552`	1735	661.4	358.0	**2.0**	**3.8**
`c6288`	2379	350.0	229.9	**2.5**	**13.6**

Table 1: Zero-delay experimental results.

Results are reported in Table 1 for a few representative benchmarks (complete tables of results are omitted for space reasons). The first two columns contain the name of the circuit and the number of gates in its *golden model*. Columns three to five report the RMSRE of constant model (Const), linear regression (Lin) and node sampling (Ns). The RMSRE was evaluated by running gate-level and RTL power simulations for input sequences with average transition probability ranging from 0.01 to 0.99. The improved robustness of node sampling is evident: the typical RMSRE is well below 10%, while the RMSRE of characterization-based models is always much greater than 100%. For benchmark circuits `c7552` and `c6288` similar results (namely, a RMSRE around 5%) were also obtained with a sampling factor of 1%.

To test the effectiveness of technology scaling, all benchmarks were re-mapped on a different technology library, containing only two-input cells with large input/output capacitances. The effect of remapping on power consumption was always in excess of 100%. Circuit `cmb` was chosen as reference benchmark $\mathcal{B}$ for technology tuning. Its *golden model* and its re-mapped version were simulated using the same input sequence (namely, a sequence of 10000 input patterns with 0.5 transition probability) to estimate P_{prov} and P_{user}, respectively. The technology scaling parameter was then computed and used to evaluate the power consumption of all re-mapped circuits (Equation 10). Results reported in the last column of Table 1 show that technology-scaling does not impair accuracy and robustness.

A second set of experiments was performed to test the quality of node sampling in full-delay power estimation. For each benchmark, a full-delay gate-level simulation was performed (after node sampling) to characterize coefficient α of Equation 9. Full-delay gate-level simulations were also used to characterize constant and linear models and to evaluate the accuracy of power estimates. Results are reported in Table 2. The RMSRE of node sampling with additional fitting coefficient is around 10%, that is more than one order of magnitude below the RMSRE of characterization-based approaches. The last column reports the accuracy of technology-scaled full-delay models. Apparently, these power models are less accurate and there is room for improvement. Unfortunately, the accuracy loss is mainly caused by the limited robustness of the linear fitting of e_S^1. Our worst result was obtained for benchmark `c6288` which is known to have high glitching activity [11].

Finally, we performed one last set of experiments to assess the complexity of model evaluation during RTL simulation. Once the node sample has been chosen, the RTL model is constructed by extracting the Boolean equations of the nodes in the sample, optimizing them by technology-independent logic optimization (using SIS [10]) and compiling the logic description into optimized C code. We do not describe this process in detail because of space

Circuit		RMS Rel. Err. (%)			
Name	Size	Const	Lin	Ns	Ns^B
cm85	31	1734.3	213.5	9.6	7.3
cm150	46	1380.1	207.6	7.4	4.9
cmb	34	1559.2	110.2	3.0	5.1
parity	36	925.5	220.9	16.3	21.5
mux	61	1303.2	124.4	3.3	3.1
alu2	252	1266.9	265.5	11.9	9.6
alu4	460	1085.8	346.2	4.7	9.4
c432	159	991.8	384.5	11.4	8.7
c880	211	1085.5	249.9	8.8	5.2
c1355	521	508.4	305.1	15.6	16.8
c7552	1735	833.0	292.1	5.3	16.1
c6288	2379	671.2	456.6	31.6	34.6

Table 2: Full-delay experimental results.

Circuit		Time (sec.)			Memory	
Name	s	Mod	Comp	Ev	Mod	Comp
cm85	5	2.8	7.3	483	1K	5K
cm150	7	3.9	16.4	833	1k	5k
cmb	5	2.6	11.1	583	1k	5k
parity	5	5.4	14.4	667	1K	7K
mux	8	7.4	18.6	1083	2K	11K
alu2	14	19.6	82.2	4740	6K	37K
alu4	24	27.4	383.8	8750	9K	69K
c432	16	17.8	48.9	2900	5K	26K
c880	23	12.7	81.5	4750	4K	37K
c1355	29	34.9	487.3	7916	9K	78K
c7552	30	62.6	3554	47166	15K	459K
c6288	42	65.5	2442	210857	18K	612K

Table 3: Evaluating CPU and memory usage. Data refer to a Sun SPARCstation 20 with 32Mb of memory.

limitations. The second column of Table 3 shows the number of nodes in the sample. Observe that the sample size grows very slowly with circuit size. This is an expected result because sample size depends on the distribution of nodal capacitances and not on circuit size. The time required to perform 1,000,000 model evaluations is reported in column 3 and compared with the time required to perform the same number of zero-delay compiled-code simulations (column 4) and real-delay event-driven simulations (column 5) at the gate level. We also report the size of the executable model in bytes (column 6) and the size of the executable code for the compiled simulation of the entire circuit (column 7). Model evaluation is 5-50 times faster than compiled-code simulation and more than two orders of magnitude faster than event-driven simulation. Memory usage is 5-30 times smaller.

6 Conclusions

In this work we have presented a methodology for RTL power modeling of IP units. Our models are automatically built without designer intervention and do not reveal details on the internal structure of the units (*i.e.*, they protect intellectual property). The most remarkable characteristic of the models is their robustness in the input statistics: the estimation accuracy is always high over a wide range of average input switching activity. This is in sharp contrast with modeling methodologies based on characterization, whose accuracy deteriorate when the unit is operated with input statistics dissimilar to those assumed during characterization.

The robustness of our method is due to the use of node sampling. A limited number of internal nodes of a gate-level implementation of the units is selected and their switching activity is used to estimate the switching activity of the entire units. The correlation between the switching activity of the node sample and that of the entire units is good and does not depend much on the input patterns that are applied to the unit.

We proposed a significant extension to the model that makes it capable of adapting to changing technologies, gate libraries and synthesis tools. Moreover, we introduced a hybrid methodology that merges node sampling and input pattern sampling to provide an accurate model of glitch power caused by non-zero gate delays.

References

[1] S. Powell and P. Chau, "Estimating power dissipation of VLSI signal processing chips: the PFA techniques," in *Proc. of Workshop on VLSI Sig. Proc.*, pp. 250–259, 1990.

[2] H. Mehta, R. M. Owens, and M. J. Irwin, "Energy characterization based on clustering," in *Proc. of Design Automation Conf.*, pp. 702–707, 1996.

[3] Q. Qiu, Q. Wu, M. Pedram, and C.-S. Ding, "Cycle-Accurate Macro-Models for RT-Level Power Analysis," in *Proc. of Int.l Symposium on Low Power Electronics and Design*, pp. 125–130, 1997.

[4] C-T. Hsieh, C-S. Ding et al., "Statistical sampling and regression estimation in power macromodeling," in *Proc. of International Conf. on Computer-Aided Design*, pp. 583–588, 1996.

[5] S. Gupta and F. Najm, "Power macromodeling for high-level power estimation," in *Proc. of Design Automation Conf.*, pp. 365–370, 1997.

[6] L. Benini, A. Bogliolo, M. Favalli, and G. De Micheli, "Regression models for behavioral power estimation," in *Proc. of PATMOS*, pp. 179–187, 1996.

[7] A. Bogliolo, L. Benini, and G. De Micheli, "Adaptive least mean square behavioral power modeling," in *Proc. of IEEE Eur. Design and Test Conf.*, 1997.

[8] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0," *Technical report, Microelectronics Center of North Carolina*, Research Triangle Park, NC, January 1991.

[9] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*, Prentice-Hall, 1982.

[10] E. M. Sentovich, K. J. Singh, C. W. Moon, H. Savoj, R. K. Brayton, A. Sangiovanni-Vincentelli, "Sequential Circuits Design Using Synthesis and Optimization," in *Proc. of IEEE Int.l Conf. on Computer Design*, pp. 328-333, 1992.

[11] L. Benini and M. Favalli and B. Riccò, "Analysis of hazard contribution to power dissipation in CMOS IC's," in *Proc. of Int.l Workshop on Low Power Design*, pp. 27-32, 1994.