Regression Models for Behavioral Power Estimation

L. BENINI,[a] A. BOGLIOLO,[a] M. FAVALLI,[b] G. DE MICHELI[a]

[a] *CSL, Stanford University, Stanford, CA 94305, USA*
[b] *DEIS, University of Bologna, 40136 Bologna, Italy*

ABSTRACT: Behavioral power estimation is required to help the designer in making important architectural choices. In this work we propose an accurate and general behavioral power modeling approach especially suited for synthesis-based design flows making use of a library of hard macros implementing behavioral operators. Power dissipation models are pre-characterized and back-annotated in a preliminary step. Accurate information on the power dissipation of the used macros can then be collected during behavioral simulation of the synthesized circuit. Our characterization and modeling methodology is based on the theory of linear regression. Optimal linear power models are obtained with methods of least squares fitting and their generalization to a recursive procedure called *tree regression*. The regression models can be used for pattern-based *dynamic power simulation* and for probabilistic *static power estimation* as well. Our behavioral simulator is integrated within PPP, a multilevel simulation engine for power estimation fully compatible with Verilog XL.

INTRODUCTION

A critical feature for the success of behavioral synthesis tools is the capability of early estimating the power dissipation of large digital systems. In this work we present a novel approach to behavioral power modeling especially suited for synthesis-based design methodologies.

In the design of large digital systems, building blocks are typically described by behavioral models. For instance, at the register transfer level (RTL) the circuit behavior is described by means of arithmetic operators and registers controlled by loop and conditional structures. RTL models are functionally accurate and enable behavioral simulation orders of magnitude faster than gate-level simulation. Since fully-functional RTL models are available generally much earlier than their gate-level counterparts, obtaining power data during RTL simulation is an attractive possibility.

Techniques for power estimation based on behavioral models have been recently proposed. While in earlier approaches (Liu and Svensson, 1994) the effect of input signals statistics was not taken into account, Landman and Rabaey (1995), proposed a technique that accounts for signal statistics and showed that power is strongly dependent on such information. Unfortunately, the applicability of this approach depends on a set of assumptions on data representation and signal statistics, and relies on human knowledge for the formulation of basic behavioral models that are subsequently automatically optimized.

Our modeling technique allows accurate power estimation in systems where the data representation and signal statistics do not satisfy the requirements for the applicability of the methods proposed in (Landman and Rabaey, 1995). We start from a library of *hard macros* implementing behavioral operators. A characterization procedure is run once for all on the library elements (for which we assume the availability of a gate or circuit level representation). We focus on hard macros implementing combinational logic primitives (arithmetic operators, steering logic, encoding/decoding logic, etc.). Sequential macros, memories, analog components are not discussed in this work.

Power models are automatically extracted and

Correspondence to: Prof. G. De Micheli.
Integrated Computer-Aided Engineering, 5(2) 95–106 (1998)

back-annotated in the behavioral representations of the library elements. The back-annotated units can be run within RTL simulation and provide a high level power estimate. Notice that characterization of hard macros can be performed once for all by the library vendor. This is not the case for soft macros, that are generated from synthesizable HDL at design time. In this work we do not deal with soft macros.

Our approach is a generalization of well-known linear regression techniques. We abstract all information on the internal structure of the unit (i.e., we assume that the circuit is a *black box*). As a consequence no human knowledge is required and the model extraction procedure is fully automatic. For a class of circuits the accuracy of the regression model can be improved if different regression equations are obtained for different modes of operations. We define a characterization procedure called *tree regression* that automatically captures this kind of behavior.

An important strength of our regression models is that they can be used for pattern-independent *static behavioral power analysis*. RTL simulation is run once for all and switching activities of the signals connecting the functional units are collected. Architectural exploration can then be performed without the need of repeating the simulation for every selection of different units that the designer wants to explore.

The experimental results show that behavioral power estimation is a feasible alternative to gate-level (or circuit level) techniques even in cases where no preliminary assumptions on data representation and signal statistics can be exploited in the pre-characterization phase. Although the loss of accuracy is sizable, our models always perform better than simple estimates based on average power.

We have embedded the behavioral power estimation tool in PPP (Bogliolo, Benini and Riccò, 1996), a multilevel power estimation engine conceived to assist the designer with accurate power information during the complete design process, from the specification to the final gate-level implementation.

PREVIOUS WORK

In the simplest kind of RTL models (Liu and Svensson, 1994; Martin and Knight, 1995), the power dissipation of a functional unit is approximated with a single fitting coefficient P, namely, the average power dissipation. The value of P is generally computed by simulating the unit with a long sequence of random input patterns possibly resembling typical input statistic. The most common assumption on the distribution of such patterns is that of *uniform white noise* (UWN). The power of a system composed by several functional units is then computed as the sum of their average power estimators.

Landman and Rabaey (1995) realized that for signal processing systems operating with 2's complement numbers, the input probability distribution is *not* UWN and the simple UWN assumption can produce large errors. They concluded that a *dual bit type* (DBT) model is a more accurate representation of signal statistics. The least significant bits have an activity pattern very close to white noise, while the most significant bits (sign bits) have high correlation and cannot be modeled as UWN. The model proposed in (Landman and Rabaey, 1995) takes input statistics into account by increasing the number of fitting coefficients to be obtained during unit characterization. An important strength of the DBT model is that it allows pattern-independent static power estimation at the behavioral level. The designer simulates the system at the RTL level and collects data on the signal statistics at the interface of the units. This information is then explcited to perform architectural power exploration: if different implementations of the functional units are available, the designer may experiment with several combinations without the need of re-simulating the RTL description.

Example 1. Consider a design where two 12-bit adders M_1 and M_2 are instantiated. Assume that four different adder implementation styles are available. The DBT power models $[s_i, u_i]$ (simplified for the sake of explanation) for the different implementation styles are respectively: $[0.5, 3.1]$, $[0.7, 4.2]$, $[0.31, 5.1]$, $[2.1, 2.7]$. The first element of each array is the power coefficient of the sign bits, while the second is the power coefficient of the UWN bits.

With a single preliminary RTL simulation (with, say, 10^5 patterns), it is found that for adder M_1 there are 7 sign bits and 5 UWN bits. For adder M_2 there are 2 sign bits and 10 UWN bits. Disregarding area and delay considerations, the first implementation is the best choice for M_1, with estimated power $P_{M_1} = .5 * 7 + 3.1 * 5 = 19$. For M_2 the best choice is the fourth implementation with estimated power $P_{M_2} = 2.1 * 2 + 2.7 * 10 = 31.2$.

This architectural choice has been made without iterating the simulation. Just one RTL simulation was required to obtain the statistics on sign and UWN bits. With pattern-dependent simulation, 16 runs of 10^5 patterns would have been required to explore all possible alternatives.

The main limitations of the DBT approach are i) the need of human knowledge for formulating basic models for the units, ii) the degradation of the accuracy to simple average power estimate when the sign bits are a small fraction of the inputs, or when the DBT model does not hold. We address both the limitations and we propose a black-box, general model that retains the desirable property of allowing static behavioral power estimation.

More recently, Mehta, Owens and Irwin (1996) proposed a behavioral characterization approach based on clustering. The methodology presented in (Mehta, Owens and Irwin, 1996) relies on the assumption that closely related input *transition vectors* (a transition vector is the concatenation of two successive input patterns) have similar power dissipation. In our experience, for many circuits the assumption is not valid. Consider, for instance, the effect of the carry-in signal on the power consumption of a full adder. Two transition vectors that differ only for the value taken by the carry-in bit in the second input pattern give rise to completely different power consumptions even if their Hamming distance is 1. Although the authors obtain an average error within 10–15% on the *same sample used for characterization*, they don't discuss the dependence of the accuracy on the input statistics. Finally, the model proposed in (Mehta, Owens and Irwin, 1996) is strongly pattern dependent: for each input pattern a table lookup must be performed to obtain the power estimate. Hence, it can be used in a pattern-based simulation context but it is not applicable to static power estimation.

LINEAR REGRESSION MODELS

Consider a functional unit with n inputs and m outputs. Assume that the circuit is stable at time t_1 and t_2 ($t_2 > t_1$), and that an input transition occurs in the time interval $T = [t_1, t_2]$. We denote by p the power consumption of the circuit in the time period T. Our goal is to find a *black-box pattern-dependent model* of p using only boundary information (i.e., the knowledge of the inputs and outputs of the unit at time t_1 and t_2).

To this purpose, we take T equal to the time period between subsequent input patterns, and we follow two simple observations: i) in a CMOS combinational circuit, some input has to switch in order to dissipate power, ii) the presence of switching outputs corresponds to some internal activity. Moreover, patterns with high input-output activity usually lead to higher power dissipation than patterns with lower activity. Obviously this is not always true also because the transitions of different signals may have a different impact on the dissipated power.

We approximate the power dissipation in the circuit by means of a linear regression model based on its input-output activity. The input (output) activity is represented by a vector of Boolean variables $\mathbf{i} = (i_1, i_2, \ldots, i_n)$ ($\mathbf{o} = (o_1, o_2, \ldots, o_m)$) taking value 1 when there is a transition on the corresponding input (output) signal. In symbols, our power estimate is

$$p = c_0 + c_1 i_1 + c_2 i_2 + \ldots + c_n i_n + c_{n+1} o_1 + c_{n+2} o_2 + \ldots + c_{n+m} o_m \quad (1)$$

where $\mathbf{c} = (c_0, c_1, \ldots, c_{n+m})$ are fitting coefficients to be determined during characterization. Notice that there are 2^n input transitions associated with the same activity vector $\mathbf{i}$.

Obviously, Equation (1) is only a rough approximation: power dissipation is also affected by several other parameters (initial input values, input slopes and signal skews) and its dependence on the I/O activity is *not* exactly linear. On the other hand, signal transitions are the main sources of power consumption and the linear power model is attractive because it is simple and it does not require any knowledge of the actual structure of the unit being modeled.

To determine the coefficients of Equation (1) we need a *sample* of input-output activities and corresponding power consumption. The sample of data collected during the characterization phase can be represented by a pair $(\mathbf{X}, p)$. If s is the sample size, $\mathbf{X}$ is an $s \times (n + m + 1)$ Boolean matrix containing the values taken by the independent variables during characterization (its k-th row being $\mathbf{x}^{(k)} = (1, i_1^{(k)}, i_2^{(k)}, \ldots, i_n^{(k)}, o_1^{(k)}, o_2^{(k)}, \ldots, o_m^{(k)})$), while $\mathbf{p}$ is a vector of size s containing the corresponding values of the dependent variable (the k-th element being $p^{(k)}$) obtained from accurate gate-level power simulation.

Given a sample $(\mathbf{X}, p)$, coefficients $\mathbf{c}$ are the unknowns of the following system of linear equations:

$$\mathbf{p} = \mathbf{X}\mathbf{c}. \quad (2)$$

Due to the statistic nature of the characterization process, the sample size must be significantly larger than the number of parameters to be characterized. Hence, matrix **X** has many more rows than columns and the linear system is over-defined. The vector c giving the minimum mean square error among all possible linear estimators of p can be obtained from (2) using well-known techniques of least squares fitting (Bowerman and O'Connell, 1990). An important property of the least squares linear model is that it always produces an estimate of p with the same average value as the average value of p in the sample used for fitting. Therefore it is guaranteed to perform at least as well as an average value approximation.

Model Validity

In this subsection we check the validity of the linear regression model by discussing the simplifying assumptions we made to construct it.

The first assumption to be checked is that there is correlation between the input-output switching activity and power dissipation. We performed several tests: a typical result is shown in Figure 1(a) where the power dissipation is plotted as a function of the total input-output activity (i.e., the number of inputs and outputs switching) for an eight-bit carry-lookahead adder. It is apparent that in this case there is good correlation between power consumption and input-output activity. This result is not general, but we experimentally found that it holds for a large set of circuits with functionality ranging from random logic to arithmetic operators. Moreover, the proposed regression model provides a deeper insight than the model used in Figure 1(a) in that it accounts for the activity of single inputs and outputs.

The second issue is the robustness of the linear model in presence of the "noise" made by the variation of parameters that do not take part in the model (such as the initial state of the input signals). An important property of the least squares equation is that it provides the optimum fit in a statistical sense. If the dependent variable can be seen as the superposition of a deterministic variable (function of the independent variables) and a random noise with Gaussian distribution, it can be shown that the least squares fit maximizes the probability that for a given value of the independent variables the dependent noisy variable has the value predicted by the least squares solution (Bowerman and O'Connell, 1990). We checked the Gaussian hypothesis by plotting the distribution of power dissipation obtained for several input transitions corresponding to the same configurations of **i** and **o** (remember that a given value of the input switching activity can be produced by 2^n different input transitions). An example probability distribution for the same adder mentioned before is shown in Figure 1(b): the bell-shaped curve closely resembles a Gaussian distribution. Again, we do not claim the generality of this result, but our tests show that it holds for a large class of circuits.

Finally, the last hypothesis to be tested is the linearity of the model. Unfortunately, power is not a linear function of the switching activity at the input and output signals. The linear model has been chosen because the theory of linear regression is well-established, and it does not require any knowledge of the internal structure of the circuit. However, trying to fit a non-linear relationship with a linear model may cause sizable errors. Moreover, the unit may have different modes of operation, with completely dissimilar power consumption.

ADVANCED REGRESSION MODELS

Although the theory of linear regression is well-established, and the procedures for model building are straightforward, the accuracy of simple regression models for power dissipation is limited by the strong non-linearity of its dependence on I/O activity. Moreover, the unit may have different modes of operation, with completely dissimilar power consumption. To address the limitations of linear regression we propose an advanced procedure closely related to the non-parametric statistical model known as *regression tree* (Breiman et al., 1993). We call our procedure *tree regression*, because it recursively builds a tree structure with linear regression equations on the leaves.

Tree Regression

The inputs of large logic units can often be grouped into two classes: *control inputs* and *data inputs*. Control inputs have very strong influence on the behavior of the units, because they select different modes of operation. On the other hand, while high activity on data inputs usually correlates well with high power dissipation, such behavior is not observed for control inputs. From this observation, it comes that control inputs can be used to *select among different regression equations*. Given a control variable, and a sample, we split the sample in two subsets, one for each value

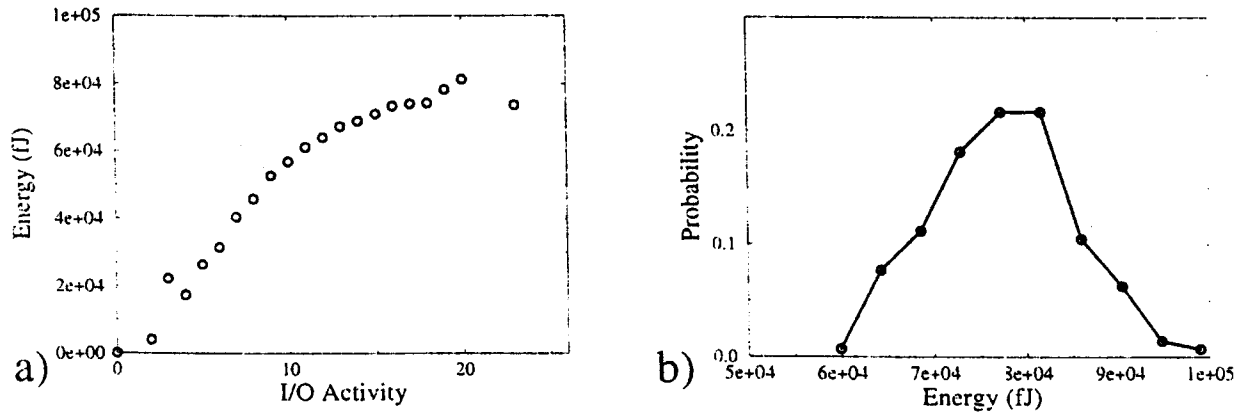

Figure 1 a) Correlation between the I/O activity of an 8-bit carry-lookahead adder and its energy-per-cycle consumption. b) Bell-shaped distribution of the energy consumption of the same circuit due to input transitions corresponding to the same activity vectors (namely, i = (0011001100011011), o = (00101001)).

of the control variable. On the two sub-samples we then compute two new linear regression models.

The advantage of this procedure is intuitively clear. If the behavior of the logic unit changes radically for different values of the control variable, a single regression model will attempt to find a linear fit between two widely spaced clusters of data. As a result, the fitting will not be satisfactory for either of the two clusters. If we split the data, and we separately fit the two clusters, much better results are obtained. The effectiveness of model splitting is illustrated in Figure 2 for a two variable function.

This reasoning can be extended to multiple control variables in a recursive fashion. Once we have split the data in two clusters, we can further split if other control variables can be found in the partial models. The structure generated by the recursive splitting is called *regression tree*. The internal nodes of the tree are labeled with the control variables on which we split the model, while the leaves correspond to regression equations with $n + m - d$ independent variables, where d is the depth of the tree. The number of leaves is exponential in the depth of the tree. Consequently, the splitting procedure must be limited to a small number of input variables.

Notice that, in principle, model splitting also addresses non-linearities. A function p of Boolean variables $x_1, x_2, \ldots, x_n$ is non-linear if and only if some of the independent variables (say x_i) affects not only the value of p, but also the dependence of p on some other variable (say x_j). In other words, x_i plays the role of a control variable. Accuracy can then be improved by using two different regression models for the two values of x_i.

Splitting Criterion

Since our goal is a black-box modeling procedure, we need an automatic splitting criterion based on boundary information. To this purpose we use a statistical approach that can be outlined as follows. i) The global regression model is computed. ii) For each independent variable x_i, the proportion of variance σ_i^2 of the dependent variable p due to x_i is computed (Bowerman and O'Connell, 1990). iii) The independent variable with the largest σ_i^2 is chosen for splitting. The rationale behind this procedure is quite simple. The variance σ_i^2 is high if a change in the value of x_i is associated to a wide variation of p (in average). In other words, if the independent variable x_i selects between two radically different behaviors of the unit, the variance of p due to x_i will be significant.

The advantage of using a statistical method to select the splitting variables is two-fold. No human knowledge is required to steer the characterization process, and the method may be also applied to units with no evident control signals, in order to isolate behaviors with good linearity characteristics. The automatic splitting process makes our regression *non-parametric*. In non-parametric regression different functional relationships are approximated with regression models for which not only the fitting parameters but also the structure of the model itself may change. The regression tree retains the soundness of linear regression, joined with the flexibility of non-parametric methods.

One last issue is the choice of the terminating conditions. If the number of samples is sufficient and the distribution of the samples is uniform, the user simply specifies the depth value, and the pro-

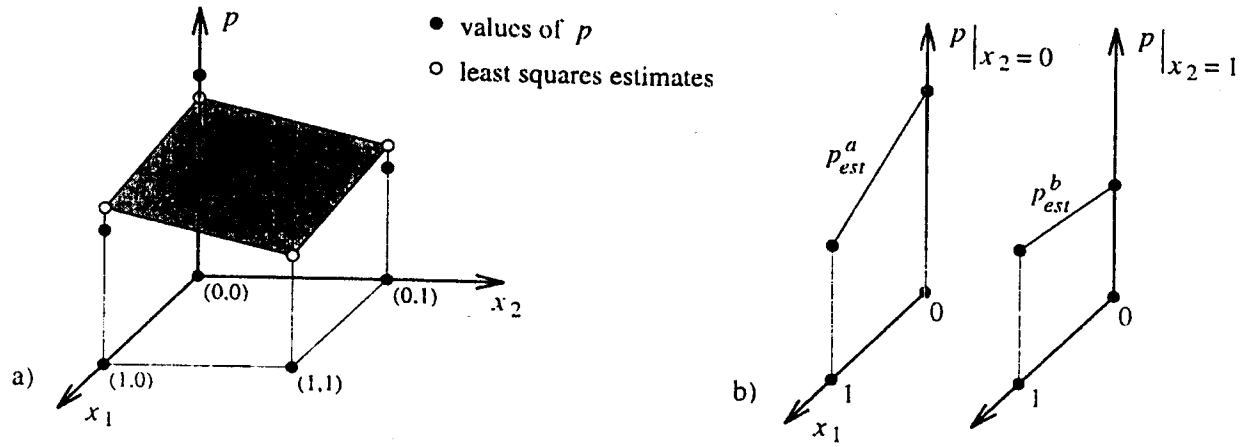

Figure 2 a) Least squares linear approximation of a non-linear function of two Boolean variables. b) Exact fitting of the same function using two linear equations of variable x_1. The value of x_2 is used to switch between the two models.

cedure automatically builds a complete regression tree. This is not always the case. Some of the independent variables are *outputs* of the module, and the user has no control on their distribution. Moreover, the input vectors may not be uniformly distributed. As a consequence, some of the branches may find singular or statistically not significant least-square matrices. In this case, the least-square equation of the level immediately above in the tree is used, with the independent variable used for the last splitting stuck at the value corresponding to the branch of the tree.

The pseudo-code of the regression tree procedure is shown in Figure 3. The procedure returns the pointer to the top node of the regression tree. The parameters are: the sample matrix **X** containing the values of the independent variables: the sample vector **p**, containing the measured values of power dissipation; the regression vector computed in the upper level of the recursions (initialized to NULL when the procedure is first called); the current level of recursion (initialized to 0); the required depth of the tree. Observe the two base cases for the recursion: least-square matrix too small (or singular) and leaf reached. The procedure `find_max_variance` selects the independent variable x_i with the maximum σ_i^2. The procedure `split`, selects the rows of **X** with fixed value $x = 0$ or $x = 1$ and the corresponding elements in **p**.

Example 2. Assume that our model has 4 independent variables x_1, x_2, x_3, x_4. We want to obtain a tree with depth 2. In the first iteration, variable x_1 is selected for splitting. The tree after the first splitting is shown in Figure 4(a). In the sub-tree with $x_1 = 1$, variable x_4 is chosen for splitting. The two leaves are obtained: they are two linear equations with 4 independent variables (Figure 4(b)).

In the subtree with $x_1 = 0$, variable x_3 is selected for splitting. The leaf with $x_3 = 1$ can be computed. The leaf $x_3 = 0$ cannot be computed because the least mean square matrix is singular. The regression equation of the parent node is then used for the leaf. The final structure of the regression tree is shown in Figure 4(c).

We have described the advantages of our method for the case of units with control inputs. It will be seen in the discussion of the results that the regression tree is useful in general. The choice of the splitting variables is exclusively based on statistical criteria, thus even units with no evident control signals may benefit of our technique that dynamically develops a model by trying to isolate behaviors with good linearity characteristics.

STATIC POWER ANALYSIS

The regression models described in the previous sections can be easily incorporated in any RTL simulator to provide *pattern-dependent* power estimates. At each clock cycle, the power consumption of the functional units is obtained from the switching activity at their I/C signals (directly available during simulation). The complexity of power evaluation is linear in the size of the model (i.e., in the number of I/O signals of each unit).

Even if power evaluation does not impair the efficiency of behavioral simulation, numerous simulations are required to obtain significant estimates of the average power. Moreover, power estimation

```
tree RegTree(X, p, c_up, level, depth) {
  if ( insufficient_sample(X,p) ) {                    /* Base case: singular matrix */
    Node→leaf = c_up;
    return(Node);
  } else {
    c = compute_least_square(X,p);
  } if (level == depth) {                              /* Base case: leaf reached */
    Node→leaf = c;
  } else {                                  /* Recursion: split on the var. with max. p-variance */
    x = find_max_variance(X, p);
    Node→splitvar = x;
    (X_then, p_then) = split(X, p, x = 1);
    Res = RegTree(X_then, p_then, c, level+1, depth);
    Node→then = Res;
    (X_else, p_else) = split(X, p, x = 0);
    Res = RegTree(X_else, p_else, c, level-1, depth);
    Node→else = Res;
  }
  return(Node);
}
```

Figure 3 Pseudo-code of the tree regression algorithm.

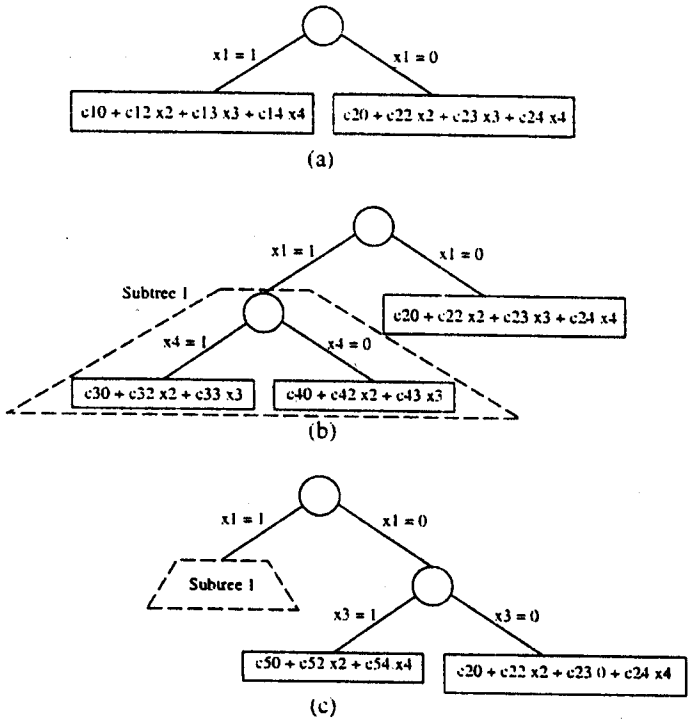

Figure 4 Steps in the construction of a regression tree with depth 2.

has to be repeated each time the designer decides to explore a different design choice by replacing one or more macros with functionally equivalent ones. As a consequence, pattern-dependent power simulation is not practical for exploring the design space. Faster (and usually less accurate) static analysis techniques are often preferred: the entire design is simulated once for all to collect statistical information about the switching activity at signals connecting functional units; signal statistics are then used to estimate power consumption.

In this section we show how the regression models we propose can be used to perform static power analysis at behavioral level, without loosing accuracy with respect to dynamic simulations based on the same models. Consider the linear model of Equation (1) applied to a n-input, m-output macro. To estimate the power consumption of an instance of the macro used in the context of a larger design, we simulate the design for a large number

of input patterns (say, N), we evaluate Equation (1) at each clock cycle and we compute the average of the N values we obtain. In symbols:

$$\begin{aligned} p_{avg} &= \frac{1}{N}\sum_{k=1}^{N} p^{(k)} \\ &= \frac{1}{N}\sum_{k=1}^{N}(c_0 + c_1 i_1^{(k)} + \ldots \\ &\quad + c_n i_n^{(k)} + c_{n+1} o_1^{(k)} + \ldots + c_{n+m} o_m^{(k)}) \end{aligned} \quad (3)$$

where apex k is used to denote the k-th clock cycle. For linearity, Equation. (3) can be rewritten as

$$\begin{aligned} p_{avg} &= c_0 + c_1 \frac{1}{N}\sum_{k=1}^{N} i_1^{(k)} + \ldots \\ &\quad + c_n \frac{1}{N}\sum_{k=1}^{N} i_n^{(k)} + c_{n-1}\frac{1}{N}\sum_{k=1}^{N} o_1^{(k)} + \ldots \\ &\quad + c_{n+m}\frac{1}{N}\sum_{k=1}^{N} o_m^{(k)} \\ &= c_0 + c_1 \tau_{i_1} + \ldots \\ &\quad + c_n \tau_{i_n} + c_{n+1}\tau_{o_1} + c_{n+m}\tau_{o_m} \end{aligned} \quad (4)$$

where the τ's are the transition probabilities at the inputs and outputs of the unit.

Equation (4) actually provides a behavioral model for *static power analysis*. Transition probabilities at the interconnections between the functional units can be computed once for all and then used to evaluate the power consumption of each element. If different macros are available to implement each functional unit, different solutions can be compared without re-simulating the circuit.

Notice that Equation (3) and Equation (4) represent exactly the same model. There is no loss of accuracy in using the static approach instead of the dynamic one. If the same set of N test patterns is used both to perform pattern-dependent power simulation and to compute transition probabilities, the two equations return exactly the same value. In both cases, the accuracy depends on the modeling assumptions discussed in the previous section and on the number of test vectors applied (N).

Similar considerations can be applied to regression trees, but in this case some accuracy may be lost when using the static power estimation model. The loss of accuracy is due to the inherent non-linearity of the regression tree. This statement can be clarified through an example.

Example 3. Consider the simple regression tree of Figure 4(a). The average power estimate provided by the model is

$$\begin{aligned} p_{avg} = \frac{1}{N}\sum_{k=1}^{N}(x_1^{(k)}(c_{10} + c_{12}x_2^{(k)} + c_{13}x_3^{(k)} \\ + c_{14}x_4^{(k)}) + (1 - x_1^{(k)})(c_{20} + c_{22}x_2^{(k)} \\ + c_{23}x_3^{(k)} + c_{24}x_4^{(k)})). \end{aligned} \quad (5)$$

By applying the same transformations used in Equation (4), we re-express p_{avg} in terms of transition probabilities:

$$\begin{aligned} p_{avg} &= \tau_1(c_{10} + c_{12}\tau_2 + c_{13}\tau_3 + c_{14}\tau_4) \\ &\quad + (1 - \tau_1)(c_{20} + c_{22}\tau_2 + c_{23}\tau_3 + c_{24}\tau_4). \end{aligned} \quad (6)$$

In this case, however, a further assumption is required to state the equivalence of the two expressions. In fact, Equation (5) is not linear, therefore the static power estimate does not imply loss of accuracy with respect to the dynamic power estimate only if x_1 is statistically independent from all other independent variables.

The pseudo-code of an algorithm for the static evaluation of a regression tree is shown in Figure 5. The initial inputs are a pointer to the root of the tree and the array T of input/output transition probabilities. Power consumption is recursively computed at each node during a depth first traversal of the tree. There are two main cases: at a leaf node the value taken by the corresponding linear equation is returned (Equation 4), while at a non-leaf node the return value is the weighted sum of those of the two branches (the weight being the transition probability τ_i of the splitting variable x_i).

EXPERIMENTAL RESULTS

We tested our methodology on a set of benchmarks from the MCNC suite and on arithmetic units generated with Synopsys' DesignWare. Notice that even if we have some partial knowledge about the benchmark interface and size, we do not know their internal structure (often we do not even know their functionality). This is the ideal testing environment for our procedure: we want to automatically generate power models for library units without using any knowledge on their structure.

The data on power dissipation has been obtained with PPP (Bogliolo, Benini and Riccò, 1996), an

```
float EvalRegTree(Node, T) {
  if ( Node→splitvar == NULL ) {                    /* Base case: leaf node */
    p = EvalLinEq(Node→leaf, T);
  } else {                                          /* Recursion: internal node */
    p_then = EvalRegTree(Node→then, T);
    p_else = EvalRegTree(Node→else, T);
    r_i = Node→splitvar;
    p = r_i p_then + (1 − r_i) p_else;
  }
  return (p);
}
```

Figure 5 Pseudo-code of the algorithm for the static evaluation of a regression tree.

Table I Results and comparison for different behavioral power models

Circuit			Const. Avg.		Lin. Reg.		Reg. Tree 1		Reg. Tree 2	
name	Ins.	Outs	RMSE	AVGE	RMSE	AVGE	RMSE	AVGE	RMSE	AVGE
alu2	10	6	0.441		0.346		0.335		0.291	
			1.154	0.903	0.484	0.138	0.501	0.197	0.510	0.180
alu4	14	8	0.388		0.294		0.275		0.260	
			1.042	0.762	0.518	0.072	0.549	0.147	0.521	0.119
c17	5	2	0.701		0.422		0.378		0.376	
			1.786	1.325	0.695	0.111	0.686	0.015	0.660	0.070
c132	36	7	0.365		0.207		0.199		0.191	
			1.128	0.849	0.390	0.086	0.385	0.071	0.403	0.122
count	35	16	0.337		0.232		0.227		0.221	
			1.362	1.181	0.428	0.136	0.421	0.103	0.401	0.073
decod	5	16	0.607		0.374		0.315		0.301	
			1.683	1.231	0.636	0.107	0.549	0.100	0.458	0.031
parity	16	1	0.204		0.174		0.164		0.163	
			0.693	0.570	0.382	0.224	0.397	0.251	0.405	0.266
pcle	19	9	0.442		0.364		0.344		0.323	
			1.307	1.038	0.602	0.136	0.605	0.178	0.578	0.113
fastdiv	17	9	0.462		0.364		0.333		0.331	
			1.193	0.729	0.677	0.050	0.666	0.076	0.685	0.086
mult	17	16	0.287		0.263		0.257		0.251	
			0.781	0.596	0.463	0.164	0.459	0.162	0.445	0.132
sqrt	9	4	0.366		0.272		0.269		0.255	
			1.110	0.807	0.496	0.053	0.507	0.112	0.510	0.121

accurate gate-level power simulator that has been reported to produce estimates within 5% from electric simulation (for library-based design in CMOS technology). Notice that electric simulation of our benchmarks would have required an excessively large amount of computation time, thus the availability of a fast and accurate power simulation tool is paramount for model building.

For each benchmark circuit we generated a large sample of input patterns and corresponding power dissipation. The input patterns used for model building were uniformly distributed and independent. In a different design environment, typical usage trace could be used. We built the regression model using linear regression and tree regression with depth one and two (i.e., two and four leaves). The regression models were then compared to the constant estimator given by the average power on the sample (i.e., a pattern-insensitive estimate equal to the average value).

Two different error measures have been used: the relative root mean square error $RMSE$ ($RMSE = \sqrt{MSE}/AVG$, where AVG is the average power on the test sample) and $AVGE$, the relative error on the average ($AVGE = |AVG_{model} - AVG|/AVG$). While $RMSE$ provides information on how well the pattern dependence of power dissipation is modeled, $AVGE$ is a measure of the accuracy in the estimation of the average power.

The results are shown in Table I. First, we estimated the accuracy of the models on a test sample composed by input vectors randomly chosen from the large sample used for characterization. In this case only the $RMSE$ is significant, because all models give by construction the same (correct)

Table II Experimental results on the 19 units in the design of Figure 6. The last three columns report the relative errors made by linear regressions (LR), regression trees of depth 1 (RT1) and static power analysis (SPA) against accurate gate-level simulations (GLS). The cycle time was of 100ns

Unit					GLS	Relative Error (%)		
Macro	n	m	Gates	Instance	(μW)	LR	RT1	SPA
CMPXX-11	22	1	48	AZero	368	16.5	10.9	9.5
				AMax	339	9.3	5.3	6.9
				BZero	372	16.3	12.2	11.0
				BMax	337	9.4	4.1	3.8
				EQCompare	449	5.5	3.5	4.0
				ResultZero1	704	35.1	28.1	28.6
				ResultZero2	686	27.9	21.9	22.0
				ResultMax1	651	31.2	21.5	23.2
				ResultMax2	640	25.4	20.8	21.0
CMPGT-11	24	2	45	GTCompare	448	12.9	10.1	10.0
EXPSBS	25	13	110	SubAminusB	1178	9.1	1.2	3.7
				SubBminusA	1216	9.0	1.6	2.9
INC-11	12	12	51	ExpAdj1	729	17.2	38.5	39.2
				ExpInc1	585	16.3	16.3	16.3
				ExpInc2	842	10.4	10.4	10.4
MUX21-11	24	11	48	ExpFirstNormSelect	742	22.2	6.8	7.6
				ExpDiffSel	392	18.0	44.1	43.3
				ExpTopSel	546	12.7	9.8	11.4
MUX51-11	60	11	118	ExpFinalSelect	596	23.9	17.5	18.1
Entire Exponent Logic					11820	15.2	7.7	8.5

average power estimate ($AVGE$ is always zero). It can be seen that the regression tree approach leads to models with improved quality compared to linear regression and constant models. Moreover, the $RMSE$ decreases when we increase the depth of the tree.

To test the flexibility of the regression models, we generated a new set of input vectors with completely different statistical characteristics from those used for characterization: the switching activity was much reduced (from 0.5 to 0.2) and some correlation was randomly introduced between inputs. In this case both $RMSE$ and $AVGE$ are significant. The performance of the constant model is unacceptably degraded, both in average and instantaneous power estimate. In contrast, the performance and robustness of the linear regression models for average power estimation is generally good. Unfortunately, the $RMSE$ is quite high, proving that linear models do not perform well as instantaneous power estimators.

Although it is clear that linear regression outperforms the simple average-power pattern-independent model, the choice between regression tree and standard linear regression is not straightforward. It seems that the regression tree is superior to linear regression when the usage patterns are similar to the characterization patterns. If this is not true, standard linear regression leads to the best results.

Finally, we tested our behavioral power estimates on a complex design of practical interest, namely, a fully-functional high-performance IEEE standard floating point adder/subtracter in double precision described in Verilog HDL. The design was composed by four units: the mantissa datapath (53 bit wide), the exponent datapath (11 bit wide), the rounding logic and the control logic (to set the various rounding modes and to select floating point sum or subtraction). The adder was designed to perform an addition/subtraction per clock cycle. The design was built starting from a library of hard macros. We discuss the power estimation of the exponent computation block, that was simulated *within the adder*.

We performed fully behavioral simulation of the whole floating point adder, with power estimation mode activated only for the units of the exponent block. The timing model used for the signal propagation through the units was a simple *constant delay model*. Thus, the simulation was not *cycle-based*: even if the timing model was not accurate, multiple transitions caused by propagation of signals through paths of unequal length were taken into account. We found that the pure cycle-based zero-delay simulation mode caused an unacceptable degradation in the quality of the power estimation, and we traded off some simulation perfor-

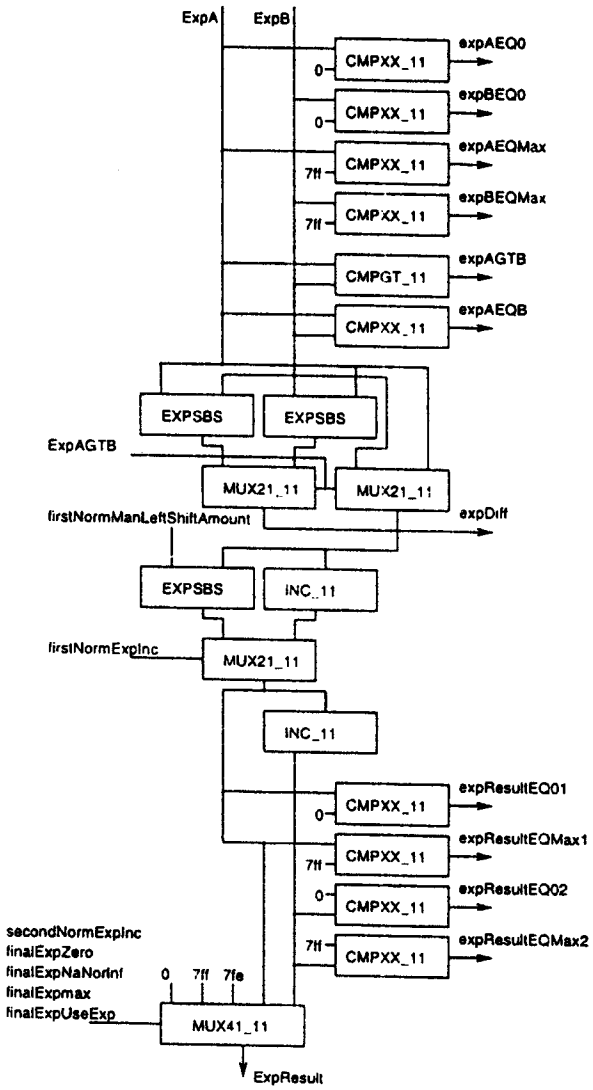

Figure 6 High-level schematic of the exponent logic of a double-precision IEEE standard floating point adder.

mance for increased accuracy in estimation.

Looking at Figure 6, we observe that several inputs to the exponent logic are controls coming from the mantissa datapath and the control logic. Additionally, the design has several internal signal reconvergences. Obviously not only the uniform white input distribution hypothesis is not valid for macros in the exponent logic, but we cannot even assume any simple distribution (such as that proposed in (Landman and Rabaey, 1995) for the numerous control inputs. Moreover, different instances of the same macro have completely different I/O statistics, depending on their location within the circuit.

The building blocks of the datapath were precharacterized before behavioral simulation. The patterns used for characterization had no relation with those provided during simulation. This is the typical usage situation: the macros are characterized once for all (possibly by the library vendor) without any detailed knowledge on where and how they will be used (by the designer). Uniformly distributed random patterns were used for characterization. Three different power estimators were compared for each of the five macro types used in the exponent logic: i) linear regression models (LR), ii) regression trees of depth 1 (RT1) used in the context of an RTL simulation, iii) static power analysis (SPA) based on the same regression trees.

Table II reports the relative errors (AVGE) for the instances of the units and for the complete block. The accuracy of the regression-based ap-

proach on the total power is evident. The compensation between over-estimates and under-estimates for the units explains why the global average error is smaller than the error on the single blocks. Notice also that for some instances of the macros the error is large, therefore it would be misleading to assume that comparisons between the estimated power consumption of different instances within the design can be made with the same degree of confidence. The range in accuracy for estimation of the power consumed by the units is due to the widely varying input statistics. Regression trees generally outperform linear regression, but sometimes they perform substantially worse (namely, for ExpAdj1 and ExpDiffSel). This is another proof of the tradeoff between accuracy and robustness that complicates the choice between linear regression and regression trees.

The last column refers to the static evaluation of the regression tree based on transition probabilities. We observe that some inacurracy is sometimes introduced by the correlation between the splitting variable and (some of) the other ones. Notice that, for the two instances of macro INC-11, there are no differences between the three power estimators. In fact, the splitting threshold criterion used to construct the regression tree was not satisfyed and the automatic characterization procedure returned a traditional regression model instead of a regression tree. As long as linear models are used, static power analysis based on transition probabilities is equivalent to dynamic evaluation performed during simulation.

Our experiments show that architectural exploration based on regression models is feasible and produces useful information. We believe that this is a key advantage of our approach when compared to table-lookup methods. Both for dynamic and static power estimation, the models have minimal computational impact.

CONCLUSIONS AND FUTURE WORK

In this work we discussed the theory and application of simple and advanced linear regression models for power dissipation of combinational hard macros. Our method does not rely on any assumption on data representation and signal probability distribution. No human knowledge is needed for providing an initial model. Our methodology is particularly well suited for design methodologies based on automatic synthesis and standard macro libraries. The linear model has limited accuracy for instantaneous power, but it is remarkably robust and sufficiently accurate for average power estimation. Regression models represent a noticeable improvement with respect to single-parameter power models and are widely applicable. Regression trees further improve the accuracy of simple linear regression by reducing the effect of non-linearities and automatically identifying control variables.

Although our method is flexible and general, it targets a specific class of circuits, namely, combinational hard macros. More work has to be done to obtain accurate black-box models of sequential macros. Our method is not suitable to model the behavior of memories. Memories are better represented by customized models exploiting their regular architecture (Landman and Rabaey, 1995).

Finally, we cannot directly model the power dissipation of controllers that are synthesized with sparse logic. In this case, our method is applicable after a fast synthesis step that provides a gate-level implementation of the controller. From the simulation of the gate level implementation, we obtain a regression model that can be back-annotated in the behavioral description to provide a first-order power estimation for the controller, that will be fully optimized in later steps of the design process.

We incorporated linear regression models and regression trees in PPP, a logic simulation engine for power estimation based on Verilog XL. PPP provides guidance to the designer during the phases of the design process, from behavioral simulation to gate-level optimization and validation.

REFERENCES

Bogliolo, A., Benini, L., and Riccò, B. (1996) "Power Estimation of Cell-Based CMOS Circuits," in *Proceedings of the Design Automation Conference*, pp. 433–438

Bowerman, B.L., and O'Connell, R.T. (1990) *Linear statistical models - An applied approach*, PWS-Kent

Breiman, L., *et al.* (1993) *Classification and Regression Trees*, Chapman and Hall

Landman, P., and Rabaey, J. (1995) "Architectural power analysis, the Dual Bit Type method," *IEEE Transaction on VLSI Systems*, vol. 3, no. 2, pp. 173–187

Liu, D., and Svensson, C. (1994) "Power consumption estimation in CMOS VLSI chips," *IEEE Journal of Solid State Circuit*, vol. 29, no. 6, pp. 663–670

Martin, R.S., and Knight, J. (1995) "Power-Profiler: optimizing ASICs power consumption at the behavioral level," in *Proceedings of Design Automation Conference*, pp. 42–47

Mehta, H., Owens, R., and Irwin, M. (1996) "Energy characterization based on clustering," in *Proceedings of the Design Automation Conference*, pp. 702–707