Lookup Table Power Macro-models for Behavioral Library Components *

M. Barocci L. Benini A. Bogliolo B. Riccó G. De Micheli †

DEIS - University of Bologna
Bologna, ITALY 40136

† CSL - Stanford University
Stanford, CA 94305

Abstract

In this work we study the problem of estimating the power dissipation of the atomic components (macros) used to implement basic operations in behavioral synthesis. We first precisely define the key requisites for behavioral power macro-models and we briefly evaluate the suitability of several approaches proposed in the past. We then focus on a specific macro-modeling methodology that appears to be best suited for usage in a behavioral synthesis tool [8]. We analyze its limitations and we propose several original improvements. Finally, we evaluate the impact of the proposed improvements on macro-model accuracy and flexibility.

1 Introduction

High-level synthesis [1] (also known as behavioral synthesis) is gaining acceptance in the design community. High-level optimization has been based on two cost metrics, namely area and speed. In the last few years, however, power dissipation has emerged as a third key cost metric. As a consequence, several research efforts have focused on the development of behavioral synthesis for low power (refer to [2] for an extensive survey). *Average power* is the cost metric targeted by these tools. In the following, we will use the word "power" as a shorthand for average power.

Behavioral-level specifications are usually provided using hardware description languages (HDLs). HDL descriptions are parsed and translated in an abstract graph-based representation called *control-data flow graph* (CDFG). Several flavors of CDFGs have been proposed in the past [1, 2]. Such representations share two common features: (i) functional dependencies are represented by arcs; (ii) operations are represented by nodes. Mapping operations to hardware resources (*resource allocation* and *binding*) and deciding when to execute the operations (*scheduling*) are the main tasks performed during behavioral synthesis.

Allocation, binding and scheduling algorithms target the minimization of a cost metric, possibly subject to constraints on other metrics. In order to manage the complexity of the optimization process, behavioral synthesis adopts an abstract view of the hardware implementation. The output of the behavioral synthesis process is a structural netlist of atomic building block called *behavioral macros*. This netlist becomes the input for lower-level synthesis tools (register-transfer level and logic level). Behavioral macros are characterized by an abstract description of input-output functionality and by cost models. They are usually collected in a *library*.

In this work we study the problem of formulating power cost models for behavioral macros (called *behavioral power macro-models* for brevity) which is a fundamental step in the implementation of high-level synthesis algorithms for low power. We first precisely define the key requisites for

This work is supported by a grant from Mentor Graphics Corporation (CoDesign Consortium)

behavioral power macro-models and we briefly evaluate the suitability of several power macro-modeling approaches proposed in the past. We then focus on a specific macro-modeling methodology that appears to be best suited for usage in a behavioral synthesis tool, namely, the look-up-table model proposed in [8]. We propose a set of original improvements for enhancing the accuracy and applicability of the basic method. We also describe a few different paradigms, spanning the tradeoff between accuracy and computational cost, for employing lookup-table models during behavioral synthesis. Finally, we present extensive experimental evidence that demostrates the practical interest of lookup table-based macro-models, and the accuracy enhancements obtained with our original techniques.

2 Power Macro-Models

Several power macro-modeling methodologies have been proposed in the recent past. The majority of these approaches targets specifically register-transfer level power estimation. In this framework, the main purpose of RTL macro-models is to provide accurate power estimates (with accuracy comparable to gate-level power simulation) for large designs with approximatively the same efficiency as functional RTL simulation (which is notoriously much faster than gate-level simulation). On the contrary, the main purpose of power macro-models in behavioral synthesis is to provide a fast estimate of power costs for comparing design alternatives.

The fundamental requisites for power macro-models for behavioral synthesis are *evaluation efficiency* and *flexibility*. The first requirement is dictated by the need of performing numerous power estimates while the synthesis tool is evaluating design alternatives. The second requirement is imposed by the fact that the same macro can be instantiated in completely different designs or operated under widely varying conditions. Furthermore, the power macro-model should be flexible enough to provide power estimates even when the environment of a macro is not completely characterized (for instance timing of input signals is unknown, fanout is only approximatively known, etc.). We focus our review on characterization-based models, i.e., models whose construction and/or tuning is based on accurate power estimates obtained by simulating/measuring the power consumed by an implementation of the macro. For these models power dissipation Pow is a function of some environmental information I (for instance, input switching activity): $Pow = \mathcal{F}(I)$. Characterization is used for finding which form of functional dependency $\mathcal{F}$ gives an accurate estimate of Pow.

The simplest macro-model is the constant *power factor* [3]: $Pow = Const$. This model has been used in several behavioral synthesis tools for low power [2]. Its main limitation is that it does not account for the strong dependency of power dissipation from input patterns, hence it can be highly inaccurate.

The power macro-models proposed by Landman et al. [4] improve upon the constant power factor by taking into account input signal statistics. Although this approach is suitable for behavioral power estimation, it has two major limitations. First, the power model for each macro must be provided by the designer (or the developer or the macro library). Second, the method relies on assumptions on the type of input signals that are valid for a class of signal-processing systems, hence it is not fully general in scope.

Regression-based macro-models [5, 6], have been proposed to overcome the above mentioned limitations. Such models do not rely on any assumption on the type of input signals and can be constructed in a fully-automated fashion. Unfortunately, they are not very robust for variations of input statistics.

The robustness issue is addressed by table-based models [7, 8]. These methods construct a lookup table addressed by some compacted form of information on the unit environment (*e.g.*, input-output switching activity and probability). They retain the desirable properties of being automatically extracted and general. Moreover, they are robust because a lookup table can represent any function with a desired accuracy, provided that the table can be made arbitrarily large. Model evaluation reduces to a simple table-lookup, flexibility and generality are satisfactory, hence table-based methods seems to be ideal candidates for implementing power macro-models for behavioral

synthesis. A fundamental issue in these methods is the selection of what type of environmental information should be considered relevant and used to address the lookup table.

When behavioral synthesis is the target, the model proposed in [8] is preferable to the model of [2]. The second model requires detailed information on input timing and probability distribution to address the lookup table. During behavioral synthesis much of this data may not be available, hence the table constructed in [2] is excessively complex.

The 3-D table model [8] postulates the following functional model for power:

$$Pow = \mathcal{F}(P_{in}, D_{in}, D_{out}) \tag{1}$$

where P_{in} is the average input probability $P_{in} = 1/N_{in} \sum_{i=1}^{N_{in}} Prob(in_i(t) = 1)$, D_{in} is the average input transition probability $D_{in} = 1/N_{in} \sum_{i=1}^{N_{in}} Prob(in_i(t) \neq in_i(t^+))$ and D_{out} is the average output transition probability $D_{out} = 1/N_{out} \sum_{i=1}^{N_{out}} Prob(out_i(t) \neq out_i(t^+))$. Function $\mathcal{F}$ is a generic function that depends only on the macro being modeled. The basic intuition of this method is to approximate function $\mathcal{F}$ with a lookup table with three indexes (i.e., a three-dimensional table), one for each controlling variable. Notice that P_{in}, D_{in} and D_{out} always lie within the interval $[0, 1]$.

The 3-D table for a macro is constructed as follows. First, n equi-spaced values are selected for P_{in} and D_{in} within the interval $[0, 1]$. Then the bound $D_{in}/2 \leq 1 - 2|P_{in} - 0.5|$ is used to eliminate unacceptable values of D_{in} [8]. For each couple (P_{in}, D_{in}), m sets of input probabilities $p_i = Prob(in_i(t) = 1)$ and transition probabilities $d_i = Prob(in_i(t) \neq in_i(t^+))$ are generated. Each set contains N_{in} input probabilities and N_{in} transition probabilities, one for each input. For a given set, the constraint on the values of input probabilities and transition probabilities is that their average is respectively equal to P_{in} and D_{in}. For each set of input probabilities and transition probabilities, k input patterns are generated. The unit is simulated with the k input patterns. Average power dissipation Pow and average output transition probability D_{out} are computed. At the end of the simulation, the cell with coordinates $(P_{in}, D_{in}, D_{out})$ of the lookup table is filled with Pow.

The process is repeated until the table has been filled. Table construction requires approximatively $k * m * n^2/2$ simulations. The size of the table is approximatively $m * n^2/2$. In the next section, we will introduce several enhancement to the 3-D table model for improving its accuracy and we discuss how it can be used during behavioral optimization.

3 Improving the 3-D Table Model

We propose two types of extensions to the basic 3-D table macro-model introduced in [8], namely extensions of the characterization procedure (i.e., table construction) and extensions of the evaluation procedure (i.e. table lookup).

3.1 Table construction

In the basic procedure illustrated in Section 2, a table entry with indexes P^*_{in}, D^*_{in}, D^*_{out} corresponds to a set of input probabilities p_i with average P^*_{in} and a set of transition probabilities d_i with average D^*_{in} for which the average output transition activity is D^*_{out}. The table entry is computed by averaging the power dissipation over k input vectors with the above mentioned p_i and d_i. To construct the table: (i) we need to generate patterns k with the required distribution using a random number generator, (ii) we need to keep track of D_{out} and (iii) we need to compute the average power.

The first task is not straightforward. Even though patterns for each input are generated with probability p_i and transition probability d_i, statistical fluctuations in the generation of the streams of zeros and ones may lead to actual values p'_i and d'_i that are slightly different from the desired ones. This in turn may lead to P'_{in} and D'_{in} different from the desired ones. If the error on P_{in} or

D_{in} is larger than half the distance between two successive samples $(1/n)$, we may actually generate a power value that should be placed in a lookup table location adjacent to the one we are targeting.

Unfortunately we have no control on the number of inputs, and forcing large values of k may lead to very long characterization times. Thus, we developed an adaptive strategy for the generation of the patterns that reduces the errors $|p_i - p_i'|$ and $|d_i - d_i'|$. The pattern generation algorithm checks, after generating a new pattern, if the values p_i', d_i' for the patterns generated so far match the desired p_i and d_i. It then biases the generation of new patterns trying to reduce the difference to zero. The bias is obtained by temporarily increasing (or decreasing) the value of p_i and d_i used for random pattern generation. We call this procedure *probability centering*.

A second problem with 3-D table construction is the lack of direct control on D_{out}. During characterization, we generate input patterns and simulate the macro with an accurate gate-level (switch level) simulator. Observing the output of the unit, we can extract D_{out}. Clearly, D_{out} depends not only on the input patterns, but also on the macro's functionality, on which we have no control. Hence, the m values of D_{out} obtained for a couple (P_{in}, D_{in}) are in general not uniformly distributed.

In the basic 3-D lookup table structure described in subsection 2, the values of D_{out} for which a power sample should be extracted are decided a priori. The lack of control on D_{out} may prevent the uniform filling of all table locations, and cause waste of memory and decreased accuracy (because the measured D_{out} would be approximated with the nearest D_{out} sampling point in the table). To address this shortcoming we organize the 3-D lookup table as a matrix of ordered lists. A matrix element is a list of m couples (D_{out}, Pow) ordered for increasing D_{out}. The matrix element is uniquely identified by a P_{in} and a D_{in}. The value of Pow is the average power (measured over k simulations) of the macro when input average probability and average transition probability are P_{in} and D_{ind}, and when the average measured output activity is D_{out}. The advantage of this data structure is that it preserves the complete information on the values of D_{out}, because their values are not decided a priori. Both the (D_{in}, P_{in}) error control technique and the list-based organization for (D_{out}, Pow) can be seen as "centering" procedures whose purpose is to guarantee that the Pow value associated with a triple $(P_{in}, D_{in}, D_{out})$ is the correct one.

3.2 Table lookup

Once the 3-D table has been constructed it can be used to estimate the power consumed by a macro. The basic power estimation procedure described in [8] is a simple table lookup. Assume that the units is instantiated in a circuit, and the values of P_{in}^*, D_{in}^* and D_{out}^* have been computed. Power consumption is estimated simply by fetching from the table the value of Pow corresponding to the measured P_{in}^*, D_{in}^* and D_{out}^*. Unfortunately the table stores only a finite number of Pow values, corresponding to a finite number of triples $(P_{in}, D_{in}, D_{out})$. If the triple $(P_{in}^*, D_{in}^*, D_{out}^*)$ is not one for which a Pow sample has been taken, we need to resort to some interpolation procedure. In [8] this problem is only mentioned, but no solution is provided.

We propose a two-step interpolation procedure based on our new table organization. First, the nearest neighbors to the couple (P_{in}^*, D_{in}^*) are found. In the most common case, there are four neighbors (corner cases with three or two neighbors are possible but will not be discussed here for the sake of conciseness): (P_{in}^4, D_{in}^4), (P_{in}^4, D_{in}^4), (P_{in}^4, D_{in}^4) and (P_{in}^4, D_{in}^4). For each neighbor, the (D_{out}, Pow) list is explored. The two nearest neighbors D_{out}^a, D_{out}^b to D_{out}^* are found, and a power value is computed by linear interpolation between Pow^a, Pow^b:

$$Pow_w = \frac{Pow^a - Pow^b}{D_{out}^a - D_{out}^b} D_{out}^* + \frac{Pow^b D_{out}^a - Pow^a D_{out}^b}{D_{out}^a - D_{out}^b} \tag{2}$$

in this way, we obtain four power values, one for each (P_{in}, D_{in}) neighbor: Pow_w^1, Pow_w^2, Pow_w^3 and Pow_w^4. The final estimated power value is obtained by *bilinear interpolation* [9]. The interpolation scheme is correct by construction on the boundaries. The complexity of power evaluation is $O(log\ m)$ if the couples (D_{out}, Pow) are ordered in D_{out}, thus, power evaluation is extremely fast.

Probability centering and interpolation try to improve the accuracy of the basic 3-D Table model. However, it is possible to take a more radical approach, and change the data structure employed in model construction. In principle, table lookup tries to select the best estimate for power dissipation on the assumption that patterns with similar P_{in}, D_{in} and D_{out} produce similar power dissipation. Thus, when performing table lookup we want to find the nearest neighbor of a given measured $(P^*_{in}, D^*_{in}, D^*_{out})$ among the $(P_{in}, D_{in}, D_{out})$ samples stored in the table. This can be performed by simply storing the samples in a list, and by scanning the list, looking for the sample that is closest to the measured triple. Efficient data structures known as *k-d trees* [10] can be exploited to perform nearest-neighbor search with expected complexity $O(log(n^2 m))$. We implemented a *k-d tree*-based method for efficient sample storage an lookup, and we compared its accuracy with the traditional lookup-table-based methods described above.

3.3 Using 3-D Tables during behavioral synthesis

In the previous section we have assumed the existence of a mechanism for computing the values of P_{in}, D_{in} and D_{out} that are required for performing table lookup and obtaining a power estimate. Obtaining these values for an instance of a macro during simulation of an RTL architecture is a straightforward task: the inputs and outputs of the macro are monitored and the probabilities and transition probabilities are computed at the end of the simulation. However, RTL simulation may be unacceptably expensive during architectural exploration. Algorithms for fast computation of P_{in}, D_{in} and D_{out} that do not require complete simulation are out of the scope of this paper (refer to [11, 12] for more information). In this subsection we simply describe how to perform power estimation with lookup table macro-models with varying amount of incomplete boundary information.

- *No boundary information.* If neither P_{in} nor D_{in} nor D_{out} are available, a rough power estimate can be computed by averaging over all Pow values in the table. Alternatively, we can use a worst-case estimate by selecting the largest Pow value.
- *Only P_{in} is available.* In this case, D_{in} can be computed, assuming no temporal correlation, by the formula $D_{in} = 2P_{in}(1 - P_{in})$. The value of Pow is computed with the interpolation algorithm described in the previous section with the only difference that the values Pow^i_w, $i = 1, \ldots, 4$ are computed by averaging over all values of Pow in the lists of the four neighbors of (P_{in}, D_{in}).
- *Both P_{in} and D_{in} are available.* Interpolation is used with the same modification as in the previous case.

Concluding this section, we stress the flexibility of the lookup table model, that can be fruitfully exploited in several ways. It can provide meaningful first-cut estimates when used within optimization loops where efficiency is key. It can also be used for more accurate and detailed power analyses, which usually take place towards the end of the optimization process, when a small number of promising alternative implementations is carefully analyzed.

4 Experimental Results

We assessed the accuracy of the lookup-table models presented in Section 3 by performing extensive experimentation on a large set of benchmark circuits. The benchmarks belong to the macro library provided with *Monet*, a behavioral synthesis tool developed by Mentor Graphics Corp. The behavioral library consists of several tens of macros, specified in VHDL, that perform typical data-processing tasks (such as addition, multiplication, division, comparison, etc.). The bit-width of the macros is parameterized (*i.e.*, it can be specified at instantiation time). It is also possible to specify the encoding of input data (sign-magnitude of two's complement).

To generate the implementation of the macros required for characterization, we exploited the `generate_lib` script provided in the Monet distribution. The script is currently used to charac-

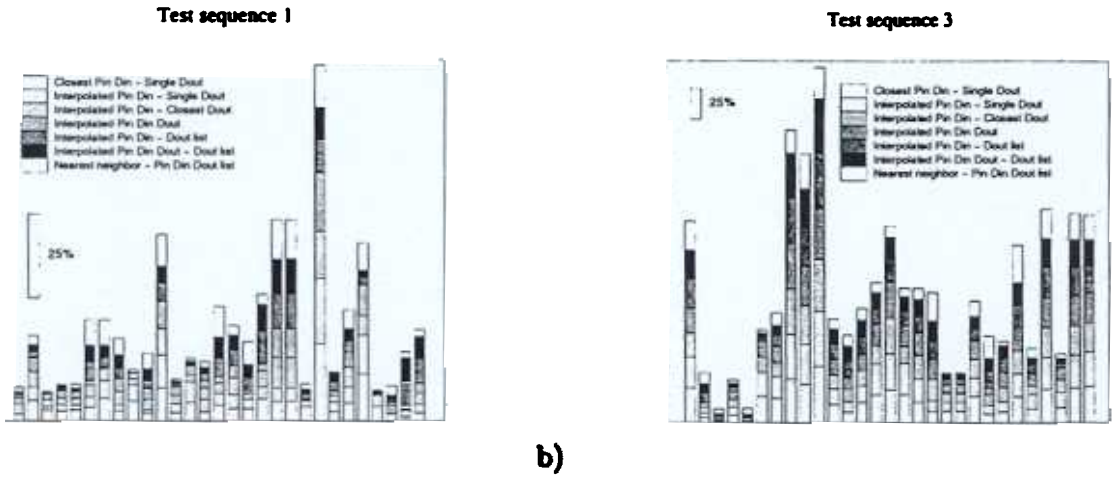

Figure 1. Comparison between the relative errors made by all the proposed methods on a set of 30 benchmark circuits. (a) Test case 1. (b) Test case 3.

terize the units for timing and area, and can be easily extended to perform characterization for power. `generate_lib` automatically instantiates several implementations for each unit (with several bit-widths and input encodings), synthesizes them using a user-specified logic synthesis tool such as Synopsys's *Design Compiler*, and finally runs the required characterization tasks to obtain information on the cost metrics of interest.

We randomly selected 30 macros instantiated by `generate_lib` and we constructed power macromodels for them. A pipelined unit was also chosen and included in the benchmark set, in order to assess the suitability of our model to deal with sequential macros. Logic synthesis was performed by *Design Compiler*. We used an accurate in-house gate-level power simulator [13] to obtain power dissipation data for characterization and validation.

We compared the accuracy of seven alternative implementations of the lookup table method. The first six use a lookup-table structure, wile the last one is based on *k-d trees*. The first method, `Closest Pin Din - Single Dout`, is the simplest one. The table has only 2 dimensions (P_{in} and D_{in}) and the *Pow* value is obtained by discretizing the given (P^*_{in}, D^*_{in}) to the nearest point in the lookup table grid. The second method, `Interpolated Pin Din - Single Dout`, is still based on a 2-D lookup table, but computes *Pow* by interpolating the values on the grid points around the measured (P^*_{in}, D^*_{in}). The third method, `Interpolated Pin Din - Closest Dout`, uses a full-blown 3-D table (with multiple *Pow* values, one for each D_{out}, for each P_{in}, D_{in} pair). First, the P_{in}, D_{in} points on the grid which are surrounding the measured (P^*_{in}, D^*_{in}) are found. For each of them, the power value *Pow* associated to the closest grid point on the D_{out} axis is taken. The final power estimate is obtained by interpolating the *Pow* values. The fourth method, `Interpolated Pin Din Dout`, uses 3-dimensional interpolation between the grid points surrounding the measured $(P^*_{in}, D^*_{in}, D^*_{out})$. The fifth and sixth methods use a list of D_{out} values for each (P_{in}, D_{in}) sample to reduce the discretization error on D_{out}, and use the same interpolation strategies as the third and fourth method, respectively. Finally, the seventh method, `Nearest neighbor - Pin Din Dout list` is a *k-d-tree based* nearest-neighbor search procedure.

We constructed a lookup table with 10 P_{in} samples, 10 D_{in} samples and 5 D_{out} sample, for a total of $10 \times 10 \times 5 = 500$ data points. To obtain the *Pow* value in each point, we averaged the results of 50 simulations. Thus, $500 \times 50 = 25,000$ patterns were simulated for characterizing each macro. Our simulator has a speed of approximatively 3,000 patterns per second per gate (on a Sun UltraSPARC), thus, characterization of a 500-gate macro requires a little more than a hour. The accuracy of the table-based model was tested against 4 different test cases. For the first test, we specified 0.5 signal and transition probability for each input. In the second test, input transition probability was 0.2 for all inputs, while input probability was 0.7. In the third test, we randomly

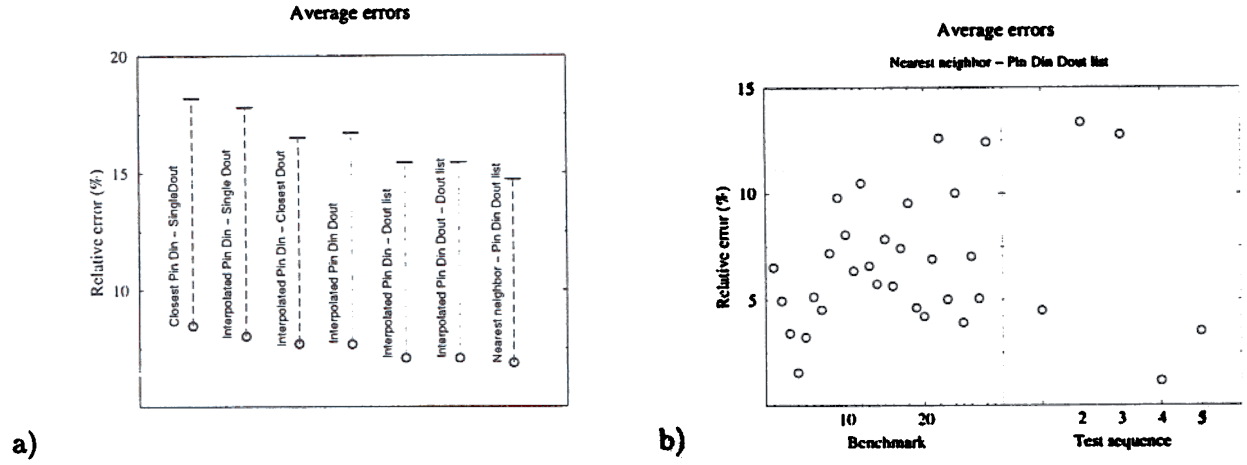

Figure 2. (a) Comparison of average percentage error for macro-models. (b) Average percentage error for the nearest neighbor macro-model.

divided the inputs in two sets: for the first set signal and transition probabilities were 0.7 and 0.2, respectively. For the second set they were 0.2 and 0.2. The last test was similar to the second, but the values of signal and transition probabilities for the two sets were swapped.

For each test sequence, we generated 20 sets of 50 patterns, we computed $(P^*_{in}, D^*_{in}, D^*_{out})$, we simulated the patterns and obtained the power dissipation P_{sim} for each set. The values $(P^*_{in}, D^*_{in}, D^*_{out})$ were used for table lookup and estimates P_{est} were obtained. The error on each estimate was computed as $\epsilon = |P_{est} - P_{sim}|/P_{sim}$, while the error for the entire test sequence was computed by averaging over the 20 values of ϵ, one for each set of 50 patterns.

The plots in Figure 1 summarize the relative performance of the various methods on the benchmarks for two different test cases (the first and the third). Each benchmark is associated with a column. The columns are divided in 7 blocks, one for each method. The height of a block is proportional to the error. Figure 1 shows that: (i) the accuracy depends on both benchmark and test case (the largest errors for different test cases are not for the same benchmark, and the errors on a given test case depend on the benchmark); (ii) the methods behave quite consistently, i.e., either they all perform well or they all perform poorly.

A global comparison among the macro-modeling methods proposed in this paper is shown in Figure 2.a. For each method, the average error (averaged on all test sequences and on all benchmarks) is shown together with its standard deviation. The average errors are respectively 8.47%, 8.01%, 7.7%, 7.68%, 7.06% and 6.84%. Standard deviations tend to slightly decrease for decreasing errors. These results lead to the following conclusions. First, using multiple D_{out} values (*i.e.*, using 3-D tables as opposed to 2-D tables) improves model accuracy. Second, interpolation is marginally useful, when applied to discretized D_{out}, but improves accuracy when applied to D_{out} lists. Finally, the noise introduced by discretization on the lookup table grid deteriorates accuracy. The best results were obtained by the nearest neighbor method that does not impose any discretization on the sample points. Furthermore, notice that interpolation does not fully compensate for the loss of accuracy due to discretization: the nearest neighbor method does not use any form of interpolation, but it is still superior to all other methods.

Figure 2.b provides a more detailed analysis of the accuracy of the nearest neighbor model. The left part of the figure shows, for each benchmark, the percentage error of the seventh method averaged over the four test cases. Errors are quite small, all below 13%. The right part of the figure shows the average error for each test sequence obtained by averaging on all benchmarks. Again, notice that errors are small, although they show a strong dependence on test sequence.

In a second set of experiments we studied the dependence of accuracy on several characteristics of the power macro-model. We selected the best method that uses a lookup table (*i.e.*, `Interpolated`

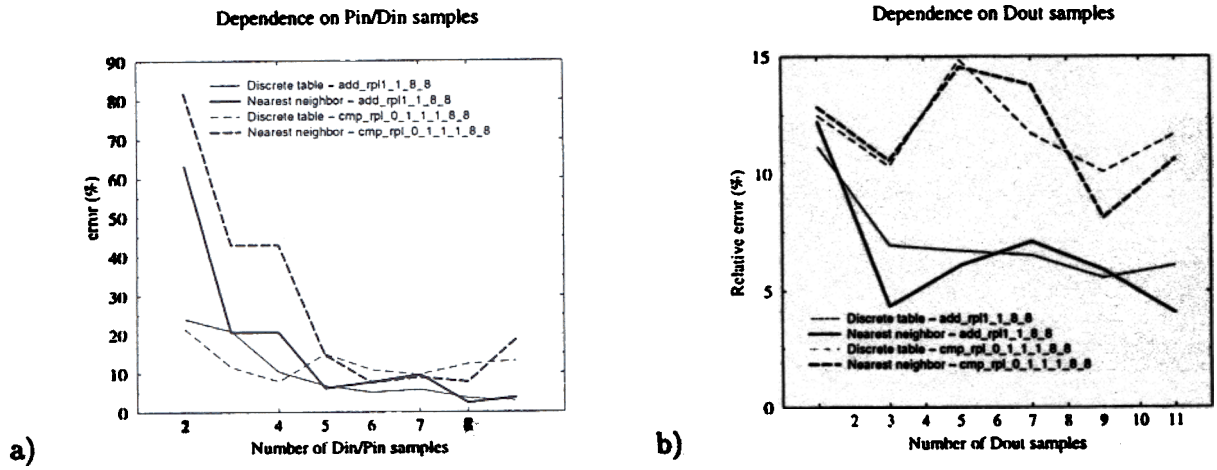

Figure 3. (a) Accuracy vs. number of P_{in}, D_{in} samples. (b) Accuracy vs. number of D_{out} samples.

`Pin Din Dout - Dout list`) and the nearest neighbor method. We also selected two benchmarks, namely a comparator and an adder. The error values were obtained by averaging over the four test cases.

In Figure 3.a we show the dependence of model accuracy on number of P_{in} and D_{in} samples. Clearly, when the number of samples is smaller than 4, accuracy is severely degraded because the lookup table becomes too sparse. On the other hand, it is not convenient to arbitrarily increase the number of samples: accuracy does not improve much, while characterization time increases linearly with the number of samples taken on any axis. The dependence of accuracy on the number of P_{out} samples (Figure 3.b) has a "noisy" behavior. Even though increasing the number of samples does help, the curves are not monotone. This is due to the fact that D_{out} values are not directly controllable and for some circuits (such as comparators) they tend to be quite insensitive to changes in input probability distribution.

Figure 4.a illustrates the dependence of relative error on the number of simulations performed to compute the Pow value for each triple $(P_{in}, D_{in}, D_{out})$. The plots show a non monotonic dependence of accuracy on number of simulations. To understand this experimental fact, remember that to compute errors we compared model estimates with average power values taken over 50 patterns. Error is low when the number of simulations used to find Pow values is close to 50, then increases when it becomes larger than 50, but not large enough to achieve statistical convergence. Finally it decreases again when the number becomes large enough to obtain statistical convergence on Pow. This result indicates that using large number of simulations to compute Pow values is the safest choice (of course, there is a tradeoff with characterization time).

Dependence on window size (*i.e*, the number of vectors that have been averaged to obtain $(P^*_{in}, D^*_{in}, D^*_{out})$ triples and their corresponding $Psim$ values to be compared with the estimates provided by the macro-models) is studied in Figure 4.b. We can infer from the plot that using small window sizes leads to better accuracy. This result is quite intuitive. Small window sizes allow us to take many measured values $(P^*_{in}, D^*_{in}, D^*_{out})$ and to obtain many estimates of Pow. Averaging over several estimates tends to improve the final accuracy level.

5 Conclusion

In this paper we have analyzed a simple lookup table-based power macro-modeling approach for providing power estimates during high-level synthesis. Extensive experimental analysis was performed to validate the model. From the experimental results we can conclude that: (i) the lookup-table model is robust and general; (ii) it can be applied to a wide class of macros, including

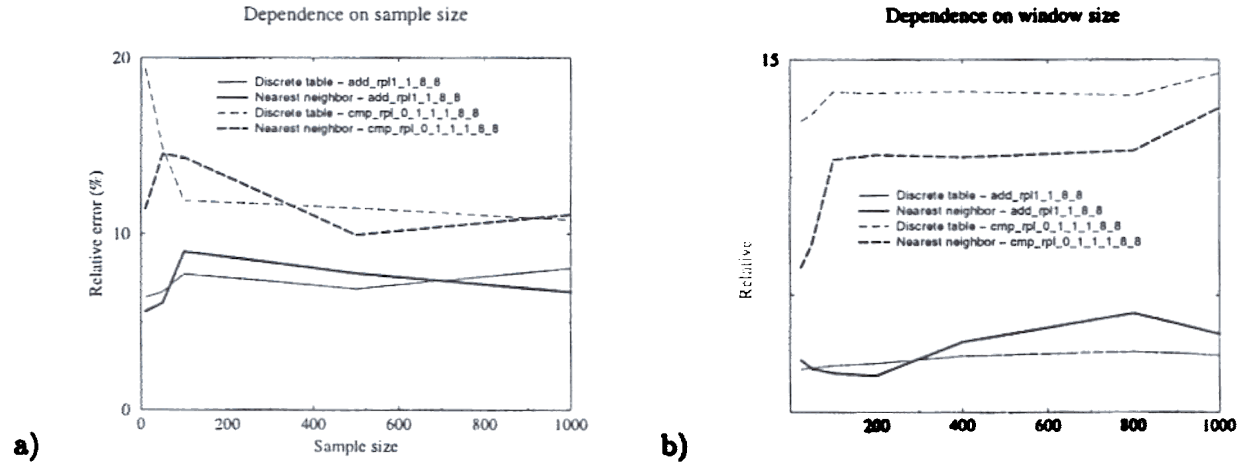

Figure 4. (a) Relative error vs. sample size. (b) Relative error vs. window size.

pipelined ones; (iii) it is easy to extract automatically and (iv) power characterization can be easily performed together with area and timing characterization.

A few interesting questions remain open. First, it could be interesting to test the applicability of the method to sequential macros with feedback. We conjecture that the presence of internal state and feedback could significantly degrade the accuracy of the method, but the results obtained on pipelined, feedback-free sequential macros are very encouraging. Second, although the accuracy on average power estimates is good (well below 10% for all methods studied), the technique can be inaccurate when input patterns are highly biased (for instance when they are deterministically selected).

References

[1] G. De Micheli. *Synthesis and optimization of digital circuits.* McGraw-Hill, 1994.

[2] A. Raghunathan, N. Jha and S. Dey, *High-level Power Analysis and Optimization.* Kluwer, 1997.

[3] S. Powell and P. Chau, "Estimating power dissipation of VLSI signal processing chips: the PFA technique," in *VLSI Signal Processing IV*, pp. 250–259, 1990.

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

[5] L. Benini, A. Bogliolo, M. Favalli and G. De Micheli, "Regression models for behavioral power estimation," *Power and Timing Modeling, Optimization and Simulation*, pp. 179–187, Sept. 1996.

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

[7] A. Raghunathan, S. Dey and N. Jha, "Register-Transfer Level Estimation Techniques for Switching Activity and Power Consumption," in *Proc. of International Conf. on Computer-Aided Design*, pp. 158–165, 1996.

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

[9] W. Press, *Numerical Recipes in C, 2nd ed.* University Press, 1992.

[10] R. Sproull, "Refinements to nearest-neighbor searching in k-dimensional trees," *Algorithmica*, no. 6, 1991.

[11] S. Ramprasad, S. Shanbhag and N. Hajj, "Analytical estimation of transition activity for DSP architectures," in *Int.l Symposium on Circuits and Systems*, pp. 1512–15, vol. 3, 1997.

[12] K. Khouri, G. Lakshminarayana and N. Jha, "IMPACT: A High-Level Synthesis System for Low Power Control-Flow Intensive Circuits," in *Design and Test in Europe Conf.*, pp. 848–854, 1998.

[13] A. Bogliolo, L. Benini, G. De Micheli and B. Riccó, "Gate-level power and current simulation of CMOS integrated circuits," *IEEE Transactions on VLSI Systems* vol. 5, no. 4, pp. 473–488, Dec. 1997.