Clustered Table-Based Macromodels for RTL Power Estimation

Roberto Corgnati Enrico Macii Massimo Poncino

Politecnico di Torino
Dip. di Automatica e Informatica
Torino, ITALY 10129

Abstract

Macromodeling *is considered the most effective approach to RTL power estimation. Among the macromodels presented in the literature,* table-based *ones have overcome some of the limitations of conventional, equation-based solutions. In this paper we propose some enhancements to the basic implementation of table-based macromodels that improve the estimation accuracy while preserving the intrinsic robustness.*

1 Introduction

The construction of abstract power models that guarantee, when evaluated, accurate power figures, is at the basis of state-of-the-art techniques for RTL power estimation. Various models have been proposed in the literature (see [1] for a survey).

Research results have shown that the so-called *bottom-up* methods generally provide the best accuracy. These methods exploit the available information on existing and/or previously designed instances of the components for which the models are being constructed. In this case, the power models are called *macromodels.*

A macromodel can be considered as a template in which power is related to a set of parameters by means of coefficients. In formula, $\mathcal{P} = \mathcal{F}(parameters, coefficients)$. The computation of the actual model (i.e., the coefficients) is carried out by tuning the model template by measuring the power consumed by several implementations of the circuit under analysis. This process, called *characterization,* evaluates the selected model template for a set of training points. Clearly, the training points should span the parameter space as much as possible, so that characterization is not too sensitive to the training set. This desirable property is defined as the *robustness* of the model.

Recently, Gupta and Najm have devised a novel, *table-based* macromodeling approach [2] that improves the robustness of conventional, *equation-based* models.

In this work, we propose an enhancement of the table-based approach that increases the estimation accuracy without affecting the intrinsic robustness of the method. This is obtained by first partitioning the input pins into two clusters, and by then augmenting the table space with two dimensions, one for each cluster. We also discuss a heuristics for the automatic generation of the clusters from a RTL specification. The results show that the accuracy of the method increases, especially for circuits with mixed data and control logic, at the expenses of a quite limited time overhead.

2 Table-Based Macromodeling

The power macromodel presented in [2] consists of building a three-dimensional look-up table of power coefficients, corresponding to the following model:

$$\mathcal{P} = \mathcal{F}(P_{in}, D_{in}, D_{out}) \quad (1)$$

where $P_{in} = \sum_{i=1}^{N_{in}} p(in_i = 1)$ is the average input probability, $D_{in} = \sum_{i=1}^{N_{in}} p(in_i(t) \neq in_i(t+1))$ is the average input transition density, and $D_{out} = \sum_{i=1}^{N_{out}} p(out_i(t) \neq out_i(t+1))$ is the average output transition density. All the three quantities are numbers between 0 and 1. This function is stored as a look-up table, with entries for each value of the three parameters.

The characterization phase, proceeds as follows. First, each dimension of the three-dimensional space is quantized into equally-spaced intervals. We denote the quantization interval as k. The values P_{in} and D_{in} are not independent, and some of the points in the 3-D grid can be dropped by enforcing the bound $D_{in}/2 \leq 1 - 2 \cdot |P_{in} - 0.5|$. Figure 1 shows the shape of the 3-D look-up table in the case of a quantization interval $k = 0.2$.

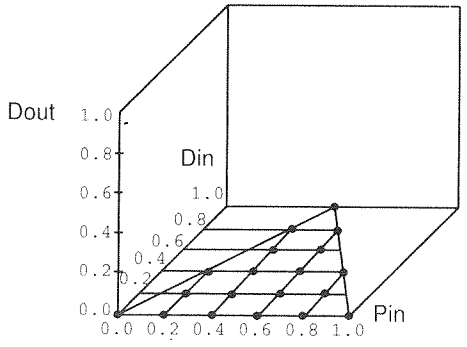

Figure 1: 3-D Table Macromodel.

For each pair (D_{in}, P_{in}), a set of input vectors is generated, with *average* input switching equal to the corresponding

0-7695-0104-4/99 $10.00

coordinate. The generation is carried out by adaptively tuning the values of input probabilities and densities in such a way that the resulting average equals the values of D_{in} and P_{in}. Each stream yields a value of D_{out} that identifies a entry in the table.

Finally, for each of the generated streams, the average power dissipation is measured with an accurate gate- or transistor-level simulator. The average of the power measurements for each of the streams is stored in the corresponding entry of the three-dimensional table.

Once the table is built for all the possible entries, power estimation simply consists of a table look-up. Clearly, the entries in the D_{out} dimension may be relatively sparse, since some values of D_{out} may never occur for some (D_{in}, P_{in}) pair. In fact, while we have full control over the values of D_{in} and P_{in}, the values of D_{out} are determined by the circuit behavior. Therefore, some entries in the table may be empty, thus requiring some intra- or extra-polation scheme in order to obtain a power value from adjacent points in the 3-D grid.

This approach guarantees a good robustness with respect to the I/O statistics of the generated streams, but its basic implementation has several potential drawbacks. First, the construction of the table requires very long characterization times. The size of the table, for a given quantization interval k, is $O(\frac{2}{k^3})$. For each point corresponding to a table entry, N streams (each consisting of M patterns) must be simulated. Overall, the total number of patterns to be simulated is approximately $N \cdot M \cdot \frac{2}{k^2}$. The values used in the original implementation are $k = 0.1$, $N = 100$ and $M = 1000$, which makes a total of $5 \cdot 10^6$ patterns per circuit. This number may require too long simulation times, especially if an accurate simulator, e.g., transistor-level, is used.

Second, the lack of control on the values of D_{out} may impair the accuracy of the method. For some circuits, very large sections of the table may be empty, making the interpolation process complicated and prone to errors.

Finally, the averaging process to obtain values of D_{in} and P_{in} has some smoothing effect on the switching properties of the individual input bits. While acceptable for data inputs, this is inaccurate for control or mode inputs, because of their specific meaning. For example, a mode control input is very likely to exhibit very long sequences of 0's and 1's, rather than continuously switching at every clock cycle. In other terms, the peculiar behavior of such bits ends up being hidden inside the dynamics of the other inputs.

Some improvements over the basic implementation of the method have already been proposed [3], with the objective of solving the first two problems. In this work, we specifically target the third observation, and we propose a method that automatically partitions the set of inputs in order to separate input bits with different switching dynamics.

3 Clustered Table-Based Macromodeling

The most intuitive way for solving the problem of the high variance of the switching dynamics for the individual inputs would be to replace the word-level quantities D_{in} and P_{in} with bit-wise ones. This would transform the model of Equation 1 into the following one:

$$\mathcal{P} = \mathcal{F}(P_{in}^1, \cdots, P_{in}^{N_{in}}, D_{in}^1, \cdots, D_{in}^{N_{in}}, D_{out}^1, \cdots, D_{out}^{N_{out}}), \quad (2)$$

where P_{in}^i, D_{in}^i, and D_{out}^i simplify now to the input signal probability, input transition probability, and output transition probability of each bit i, respectively.

Obviously, this is just a theoretical solution, since it would drastically increase the size of the table and the corresponding characterization time. Furthermore, the model size increases with increasing number of input and output pins of the circuit.

A reasonable trade-off between the two extremes of Equations 1 and 2 would be to identify a few, different levels of input activity and density, and to associate one individual dimension of the table to each of these levels. We have experimentally observed that it is sufficient to partition the input switching dynamics into *two* sets: One corresponding to input bits with *high* switching, and another for bits with *low* switching. Switching here refers to the average number of switchings over a given stream that is supposed to represent the typical external stimuli provided to the circuit. By separating these two sets, that we will call *clusters* hereafter, we can de-couple the averaging effect caused by P_{in}^i and D_{in}^i. The modified table-based model becomes now the following:

$$\mathcal{P} = \mathcal{F}(P_{in}^L, P_{in}^H, D_{in}^L, D_{in}^H, D_{out}) \quad (3)$$

where superscripts L and H refer to the high-switching and low-switching clusters, respectively. Notice that the D_{out} dimension is not partitioned in two sets, since its values are a result of simulation and cannot be controlled.

Equation 3 has now twice as many dimensions with respect to the original model of Equation 1, and we should then expect larger characterization times because of the larger number of characterization points. However, the separation of the input set into two clusters with different levels of activity profiles allows to relax the quantization of the model space.

Consider the situation depicted in Figure 2, that shows the grid corresponding to a quantization of $k = 0.1$ in the plane (D_{in}, P_{in}). Because of the averaging effect, we need to keep the grid as fine as possible, since, *a priori*, we cannot exclude any combination. On the other hand, when we split D_{in} and D_{out} in two sets, we have some degrees of freedom. For example, consider the plane (D_{in}^H, P_{in}^H); these two quantities represent the average of medium-to-high switching bits. Therefore, it is highly unlikely that a point as $(D_{in}^H = 0.1, P_{in}^H = 0.1)$ will occur very often. In other words, we are allowed to use a coarser grid in the regions representing switching levels that are different from those corresponding to the dimension of the given plane. This means that we can use a coarser grid for small D_{in}^H and P_{in}^H values and for high D_{in}^L and P_{in}^L values.

In numbers, for a quantization of $k = 0.1$, the basic method uses $\frac{2}{k^3}$ grid points, that is, 500 points. With the partitioned space, we can limit ourselves to about 6 to 9 points

per each plane, which makes a total of about 1500 points. In Figure 3 are shown two possible grid arrangements for the (D_{in}^L, P_{in}^L) and (D_{in}^H, P_{in}^H) planes.

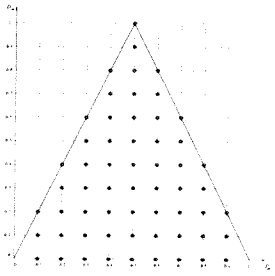

Figure 2: (D_{in}, P_{in}) Grid for the Original Method.

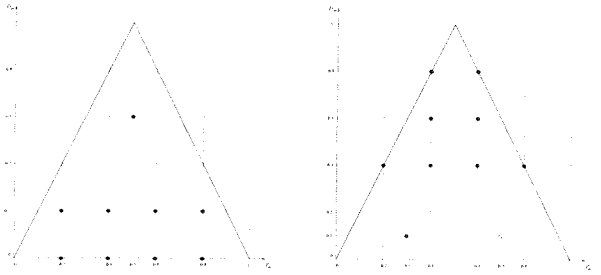

Figure 3: Grid for the Clustered Method: (D_{in}^L, P_{in}^L) Space (left), and (D_{in}^H, P_{in}^H) Space (right).

4 Cluster Selection Heuristics

There are several ways to determine a reasonable partition of the input set into clusters with high and low switching activity. The effectiveness of the various solutions depends on the type of information that is available for the design under analysis. In some cases, the information may be extracted by the RTL description, either from pin naming or by annotation of behavioral data. In other situations, when the circuit under analysis is provided with a typical input trace to be used for verification purposes, this can be used to extract individual pin switchings. In both cases, the clusters can be built by hand.

The most challenging situation occurs when the circuit is given as a black-box (e.g., an instance of a synthesizable macro), whose *functional* behavior is unknown. The only behavior that can be inferred can be obtained by RT-level simulation. This has been the case for the standard benchmark circuits (e.g., the Iscas'85 suite [4]), whose description is absolutely generic, and whose functional behavior is only marginally known. In these situations, the input pins can only be partitioned in clusters *automatically*. Ideally, this should correspond to partitioning the inputs according to their *impact* onto the power dissipation of the circuit itself. We have devised a *complexity-based* heuristic that tends to identify inputs by their impact on the *capacitive* component of power.

The proposed method simplifies the problem by distinguishing low-switching bits from high-switching ones according to their functionalities; in particular, control or mode bits correspond the low-switching set L, while data inputs identify the high-switching set H. We devised a heuristic algorithm to identify such sets of bits directly from the circuit description. The algorithm is based on implication analysis and therefore it requires some structural information about the circuit (e.g., a low-effort implementation).

The identification of clusters L and H is based on the following observation. Control bits usually trigger totally different operational modes of a circuit. Therefore, we assume that if we assign a constant logic value (0 or 1) to a control input and we propagate that constant through the network, the simplification of the circuit that we obtain is much more noticeable than that corresponding to the assignment of the same constant value to a generic data input bit. This intuitive consideration is used to drive the partitioning process. A high-level pseudo-code of the algorithm is shown in Figure 4.

```
procedure BuildInputClusters(C,α) {
1    H = L = { };
2    Size = NumNodes(C);
3    foreach (primary input x_i) {
4        C^0 = NetworkSweep(C,0);
5        C^1 = NetworkSweep(C,1);
6        Gain = max(NumNodes(C^0), NumNodes(C^1));
7        if (Gain < (α · Size)) L = L + {x_i} ;
8        else H = H + {x_i} ;
     }
     return (L,H);
}
```

Figure 4: Complexity-Based Clustering Algorithm.

The procedure `BuildInputClusters` receives, as inputs, the circuit description C and a threshold value α. The quantity *Size* is used to estimate the complexity of the original circuit (Line 2). Then, for each primary input x_i (Line 3), the two simplified circuits C^0 and C^1 are determined by imposing a constant 0 and 1, respectively, on input x_i (Lines 4 and 5), by simply propagating such values through the circuit, and by removing the so-created dangling nodes through a `sweep` operation [6]. Then, the maximum between the sizes of the simplified circuits C^0 and C^1 is stored in variable *Gain* (Line 6). If *Gain* is smaller than a specified fraction of the original size (i.e., a constant value at input x_i results in a sizable simplification of the circuit), we classify x_i as a control input, and add it to L (line 7); otherwise we add it to H (Line 8). After processing all the inputs, the two sets L and H are returned (Line 10).

5 Experimental Results

We have implemented the clustered table-based model and the clustering procedure as an extension of SIS [6]. Experiments have been run on a DEC AXP 1000/400 with 256 MB of memory. The power results for the validation of the results have been obtained with an in-house gate-level, event-driven simulator, that has also been used to collect the power values during the characterization. This choice

guarantees the consistency between actual power values and table entries.

We present two sets of experiments. The first one aims at comparing the relative quality of the conventional approach and the clustered one on the Iscas'85 benchmarks [4]. Tables 1 and 2 show the results obtained for random streams and biased streams, respectively. The power figures in the tables are obtained by averaging the results of the simulation of 20 test streams with the corresponding statistics. The first four columns describe the topological characteristics of the benchmarks, column *Table Error* reports the average and (over the 20 test streams) and the maximum estimate errors of the original table-based method [2], while column *Cluster Error* gives the same information referred to the proposed solution.

We notice that, in the case of random streams, the traditional technique performs better than the clustered one (see the data in Table 1). This was expected, since for this type of streams the additional information provided by the two clusters can not be exploited and the new solution is penalized from the coarser grid of the clustered table. On the other hand, when the streams properly exercise the circuit according to its functionality, the errors for the traditional method increase, and the comparison is sensibly in favor of the clustered method (see Table 2).

Circuit	*I*	*O*	*G*	*Table Error*		*Cluster Error*	
				Avg	*Max*	*Avg*	*Max*
c432	36	6	265	4.70	14.93	15.99	86.72
c499	41	32	525	2.49	7.22	9.12	27.75
c880	60	26	431	4.14	7.38	12.00	24.19
c1355	41	32	525	3.48	14.99	13.60	31.01
c1908	33	25	529	5.43	15.94	8.35	26.90
c2670	233	140	808	4.51	14.65	12.52	26.64
c3540	50	22	1289	4.90	20.94	13.92	42.14
c5315	178	123	1759	4.55	11.73	10.82	26.87
c6288	32	32	3240	4.02	11.25	11.12	24.47
c7552	207	108	2314	7.60	24.05	16.97	46.08
Avg.				4.58	14.31	12.44	36.28

Table 1: Results for Random Input Streams.

Circuit	*I*	*O*	*G*	*Table Error*		*Cluster Error*	
				Avg	*Max*	*Avg*	*Max*
c432	36	6	265	25.68	71.19	10.56	20.93
c499	41	32	525	7.92	22.92	2.61	8.28
c880	60	26	431	30.71	80.68	7.12	18.82
c1355	41	32	525	11.60	40.47	4.10	15.37
c1908	33	25	529	13.37	33.32	3.85	10.75
c2670	233	140	808	44.63	81.95	4.41	12.16
c3540	50	22	1289	29.66	70.19	5.52	12.79
c5315	178	123	1759	27.66	53.08	3.75	11.26
c6288	32	32	3240	10.47	18.06	5.08	10.22
c7552	207	108	2314	33.82	79.08	6.44	14.15
Avg.				23.55	55.09	5.34	13.47

Table 2: Results for Biased Input Streams.

Since it is difficult to build realistic streams for the Iscas'85 benchmarks, we have tried another experiment that allows us to use actual verification patterns for testing. We have taken a commercially available RTL-macro from the Synopsys DesignWare Foundation Library. This macro implements the function $F = a \times b + c$ onto N-bit integers. The integer representation used is determined by an additional control input TC; when $TC = 1$, two's complement is used. We have instantiated the macro with $N = 16$.

The characterization with the method described in Section 3 yields two clusters: One with the TC input and some of the most significant bits of the two data words; the other with the remaining bits. We have then fed to this circuit two meaningful input streams that are consistent with both functionality and data representation.

In the first stream *S1*, the TC input is always 0. In this case, the data are pure binary and are assumed to be purely random. In the second stream *S2*, TC is always 1. Therefore, data are coded as two's complement, and data inputs are generated with different switching profile for high- and low-order bits, according to the idea of the Dual-Bit-Type model [7], i.e., random for lower-order bits, and with lower switching in higher-order bits.

Stream	*Table Error*		*Cluster Error*	
	Avg	*Max*	*Avg*	*Max*
S1	17.34	32.12	8.42	19.10
S2	27.43	62.81	4.12	9.93

Table 3: Results for Biased Input Streams.

Table 3 shows the results that confirm that the use of a meaningful stream results in a sensibly improved accuracy.

6 Conclusions

We have proposed a modified table-based macromodeling technique that increases the estimation accuracy of the basic method without affecting its robustness. The improvement consists of increasing the dimensionality of the table by using a separate parameter for each of two clusters in which the set of the input pins is partitioned. We have also discussed a heuristics that allows the automatic generation of the clusters from an RTL specification. The results show that the accuracy of the method increases, especially in the case of circuits with both data and control, at the cost of a marginal increase in the CPU time.

References

[1] P. Landman, "High-Level Power Estimation," *ISLPED-96*, pp. 29-35, Monterey, CA, August 1996.

[2] S. Gupta, F. N. Najm, "Power Macromodeling for High-Level Power Estimation," *DAC-34*, pp. 365-370, Anaheim, CA, June 1997.

[3] M. Barocci, L. Benini, A. Bogliolo, B. Riccò, G. De Micheli, "Lookup Table Power Macromodels for Behavioral Library Components," *VOLTA-99*,, Como, Italy, March 1999, To Appear.

[4] F. Brglez, H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," *ISCAS-85*, pp. 785–794, Kyoto, Japan, June 1985.

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

[6] E. M. Sentovich, K. J. Singh, C. W. Moon, H. Savoj, R. K. Brayton, A. Sangiovanni-Vincentelli, "Sequential Circuits Design Using Synthesis and Optimization," *ICCD-92*, pp. 328-333, Cambridge, MA, October 1992.

[7] P. Landman, J. Rabaey, "Architectural Power Analysis: The Dual-Bit Type Method," *IEEE Transactions on VLSI Systems*, Vol. 3, No. 2, pp. 173-187, June 1995.