A Multi-Model Power Estimation Engine for Accuracy Optimization

Felipe Klein, G. Araujo, Rodolfo Azevedo
Institute of Computing, UNICAMP, Brazil
{klein, guido, rodofo}@ic.unicamp.br

Roberto Leao, Luiz C. V. dos Santos
Computer Sciences Department, UFSC, Brazil
{leao, santos}@inf.ufsc.br

ABSTRACT

RTL power macromodeling is a mature research topic with a variety of equation and table-based approaches. Despite its maturity, macromodeling is not yet widely accepted as an industrial de facto standard for power estimation at the RT level. Each approach has many variants depending upon the parameters chosen to capture power variation. Every macromodeling technique has some intrinsic limitation affecting either its performance or its accuracy. Therefore, alternative macromodeling methods can be envisaged as part of a power modeling toolkit from which the most suitable method for a given component should be automatically selected. This paper describes a new multi-model power estimation engine that selects the macromodeling technique leading to the least estimation error for a given system component depending on the properties of its input-vector stream. A proper selection function is built after component characterization and used during estimation. Experimental results show that our multi-model engine improves the robustness of power analysis with negligible usage overhead. Accuracy becomes 3 times better on average, as compared to conventional single-model estimators, while the overall maximum estimation error is divided by 8.

Categories and Subject Descriptors
C.4 [Performance of Systems]: Measurement techniques

General Terms: Design, Experimentation, Measurement

Keywords: Low power design, Power estimation, Power macromodeling, SystemC, PowerSC

1. INTRODUCTION

The offer of increasing amounts of ever-shrinking transistors and the demand for battery-powered mobile devices have made low-power consumption one of the most important concerns in contemporary VLSI design. Power modeling is the keystone for the whole building of power-conscious design tools.

Power macromodeling is considered today the state-of-the-art paradigm for power estimation at the RT level [13], with a variety of equation and table-based approaches [1,3–9,11]. Each approach has many variants depending upon the parameters chosen to capture power variation (e.g. signal probability, transition density, spatial correlation).

Despite its maturity, macromodeling is not yet widely accepted as an industrial de facto standard for power estimation at the RT level. This paper claims that one of the main reasons impairing its widespread use is that each macromodeling technique makes assumptions that lead to some sort of intrinsic limitation, thereby affecting its accuracy. In [7], for instance, signal statistics are assumed as uniformly distributed among all inputs and outputs, although input signal imbalances may result in significant errors [12]; the training set is limited to a single input stream per characterized point in [1], even though the choice of a proper set is crucial to obtain high-quality models [16].

Most power estimation tools rely on a single macromodeling technique. However, given a system component, there might exist an alternative technique leading to a more accurate estimate. Therefore, the proper selection of an alternative from a kit of supported macromodels could be envisaged as a way of optimizing the overall power estimation accuracy.

Although a related work [2] reports the use of multiple macromodels to improve power assessment performance, to our knowledge, no approach has been proposed to address the impact of multiple techniques on power evaluation accuracy.

This motivated us to develop a new multi-model power estimation engine that selects the macromodeling technique leading to the least estimation error for each system component and for each input-vector stream. A proper selection function is built immediately after component characterization and used during estimation. Given an input-vector stream stimulating an RTL component, the selection function chooses the most suitable macromodel for a system component according to the input stream’s signal probability and transition density.

Our engine relies on a pre-existent framework called PowerSC [10], which is a SystemC [14] extension obtained by adding power-aware C++ classes.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 provides evidences that single-model estimation approaches may incur significant errors. Section 4 describes how our multi-model engine works to improve accuracy. Section 5 shows experimental results comparing the accuracy obtained with our engine to that of conventional approaches. Our conclusions are drawn in Section 6.
2. RELATED WORK

Most of the techniques used in today’s tools are based on the concept of macromodel [11, 15], which is an abstract model obtained by measuring the power consumption of existing implementations with the help of lower level methods. Essentially, the macromodeling techniques can be divided into three categories: equation-based, table-based and hybrid.

Most equation-based techniques make use of statistical methods for model building (e.g., regression analysis). For instance, least square estimation is employed in [4] to derive the fitting coefficients. One advantage of statistical approaches is that they inherit a solid mathematical foundation and are therefore capable of providing a good level of confidence for the estimates, if adequate training sets are used. Obviously, the more an input stream is similar to the training streams, the higher the confidence level. Many other techniques belonging to this category can be found in the literature [6, 9].

The first table-based technique was proposed in [7]. Basically, it consists of a three-dimensional table, where the axes represent input/output signal statistics, and a table entry represents a power value. The original technique was improved by including a new parameter in the model, resulting in a four-dimensional table [8]. Several other works belonging to this category were proposed in the literature [3, 5].

The technique proposed in [1] lies in the hybrid category. It consists of a two-dimensional table, which is indexed by transition density and signal probability values. For each entry in the table, an equation is built by means of linear regression.

A recent work [2] describes a multi-model approach for SoC power analysis aiming at improving power simulation performance. This approach is based upon multiple models at distinct levels of granularity. The coarser the granularity, the less parameters are needed for estimation, thereby reducing the estimate overhead. The finer the granularity, the higher the accuracy, but the higher the overhead. A model is dynamically chosen for a component according to its relative power contribution to the overall system power. In other words, if a component doesn’t contribute much to the system consumption, a coarser model is selected since a higher error can be accommodated.

As opposed to that approach, we envisage to exploit multiple models to get better accuracy. Our motivation is based upon the fact that most macromodel assumptions lead to limitations, as discussed in the next section.

In brief, we propose an approach that takes a set of circuit components at the same level of abstraction and chooses the best candidate macromodel for each one, based on a prior macromodel evaluation, so as to reduce the estimation error.

3. SINGLE-MODEL LIMITATIONS

The quality of a macromodel is strongly dependent on the training set. For instance, since the macromodel in [7] assumes that signal statistics are uniformly distributed among inputs and outputs, input signal imbalances may result in significant errors as pointed out in [12]. Furthermore, the macromodel proposed in [1] restricts the training set to a single input stream per characterized point. However, the choice of a proper set is crucial to obtain a high-quality model. As shown in [16], while some training sets lead to high-quality models (avg. errors around 6%), others result in unacceptably poor ones (avg. errors around 660%).

The sensibility to the training set and some inherent macromodeling assumptions (e.g., uniform probability distribution [7], lack of modeling for output signal properties [9]) lead to intrinsic limitations that degrade estimation accuracy.

This section provides the reader with some preliminary evidences that macromodel-based estimation may incur unacceptably high errors and then pinpoints the sources of inaccuracy by means of a few examples to justify the claim that no macromodeling is robust enough to be employed alone. Therefore, this section paves the way to Section 4, where we show how our engine can automatically handle multiple macromodels to improve the overall estimation accuracy.

3.1 Macromodeling Background

For the sake of proof of concept, three different macromodeling techniques were adopted, without loss of generality, to be implemented in our multi-model engine. From now on, they will be referred to as 3DTab [7], EqTab [1] and e-HD [9]. We summarized below some background on the selected techniques, such as key estimation concepts and main characterization steps, as a base for the discussions in Sections 3.2 and 3.3:

- **3DTab**: It relies on the following signal properties: the input signal probability $P_{in}$, the input transition density $D_{in}$, and the output transition density $D_{out}$. The model is represented by a 3D look-up table such that to each valid triple $(P_{in}, D_{in}, D_{out})$ corresponds a power value. Power estimation consists in first running an RTL simulation to collect the signal statistics $P_{in}$, $D_{in}$ and $D_{out}$ and then looking up in the table for the corresponding power value. When the collected signal statistics don’t directly correspond to a valid triple, interpolation is used to return the closest value.

- **EqTab**: It considers the individual contribution of each input/output bit position. This technique relies on a look-up table which is indexed with $(P_{in}, D_{in})$. For each entry in this table, instead of directly storing a power value, the corresponding entry actually stores the coefficients of an equation.

- **e-HD**: It relies on an equation which expresses power as a function of two distinct input signal properties: the Hamming distance $h(u, v)$ and the number of stable bit ones $s(u, v)$ between two successive input vectors $u$ and $v$. The technique doesn’t employ output signals. As opposed to the previous techniques, it calculates the power on a per-cycle basis.

3.2 The Conventional Single-Model Approach

To illustrate how estimation accuracy can largely diverge among power models, we employ two real-life components as examples (*Add*, *ECLA32* and *MulS16*). Both examples were characterized for the three macromodeling techniques summarized in Section 3.1 (3DTab, EqTab and e-HD).

An experiment was conducted to assess the average errors of each macromodel as a function of two input-stream parameters, namely, $P_{in}$ and $D_{in}$, using 0.1 as the discretization step. A set of 5000 streams was generated to properly cover the $P_{in}$ x $D_{in}$ input space. Then, the RTL macromodel estimates were compared to the gate-level estimates, for each pair $(P_{in}, D_{in})$. 

The quality of a macromodel is strongly dependent on the training set. For instance, since the macromodel in [7] assumes that signal statistics are uniformly distributed among inputs and outputs, input signal imbalances may result in significant errors as pointed out in [12]. Furthermore, the macromodel proposed in [1] restricts the training set to a single input stream per characterized point. However, the choice of a proper set is crucial to obtain a high-quality model. As shown in [16], while some training sets lead to high-quality models (avg. errors around 6%), others result in unacceptably poor ones (avg. errors around 660%).
Figures 1, 2 and 3 show the error distribution on the $P_{in} \times D_{in}$ space for both examples, and for the three adopted techniques.

![Graphs showing error distribution for different techniques](image)

**Figure 1: Error Distribution (3DTab)**

**Figure 2: Error Distribution (EqTab)**

**Figure 3: Error Distribution (e-HD)**

Notice that, for a given component, the distinct macromodels lead to quite different average errors. For instance, regardless of the chosen component, the e-HD macromodel leads to the highest average error for the input streams whose transition density is in the interval $[0,0.6]$. This means that 3DTab or EqTab would be a better choice for such streams. However, for the streams whose transition density is greater than 0.6, e-HD exhibits the highest accuracy.

### 3.3 Analysis of Macromodeling Limitations

Each macromodel has its merits in capturing power variation. However, each technique implies a different usage of parameters, which eventually hides some assumption. This subsection shows that such implicit assumptions are the very source of limitation and that they are difficult to overcome within the scope of a single macromodeling technique.

Since it has been demonstrated in [8] that there is a mathematical relation between $P_{in}$ and $D_{in}$, to illustrate the drawbacks of the selected macromodeling techniques, we use the transition density ($D_{in}$) only, for the sake of simplicity.

Let’s first consider the 3DTab technique. It assumes that $P_{in}$, $D_{in}$ and $D_{out}$ are uniformly distributed through all input/output signals, although some signals could have a higher impact on the power consumption than others. Such assumption is clearly inadequate when irregularly structured circuits are addressed or when control signals are considered, since they may completely change the circuit’s operational mode [5].

To illustrate that such assumption leads to a drawback, consider again the example MulS16 (a 16-bit multiplier used as a component of the stereo audio crossover design to be discussed in Section 5). We simulated one of its implementations with 100 distinct input streams and monitored one of its operands (16 inputs). For most streams, we observed that only four of the monitored inputs exhibited transition densities higher than zero.

Let’s now consider one of those simulation instances, whose overall input transition density is $D_{in} = 0.0912$ and whose bit-wise transition densities are the following:

$$(D_0, D_1, ..., D_{15}) = (0.49, 0.44, 0.50, 0, 0, ..., 0, 0.33).$$

Notice that only the last and the first three inputs actually switch for this stream. This means that part of the MulS16 circuit is not stimulated, as opposed to the uniform distribution assumed by 3DTab, which implies the following bit-wise transition densities:

$$(D_0, D_1, ..., D_{15}) = (0.0057, 0.0057, ..., 0.0057).$$

We can therefore conclude that, despite the same $D_{in}$, such rather different stimulation patterns are likely to lead to quite different power estimates.

EqTab overcomes this drawback by taking each input into account individually, which in principle should lead to better accuracy. However, as opposed to 3DTab, a single stream is used in the characterization process for each entry in the table. Since the number of possible streams grows exponentially with the input width for some chosen input statistics, this assumption represents a serious limiting factor, as illustrated by the following example.

Let $A$ and $B$ be 4-bit input operands of a Booth multiplier. Consider a candidate characterization stream whose overall transition density is $D_{in} = 0.25$ and whose bit-wise transition densities are:

$$(A_0, ..., A_3, B_0, ..., B_3) = (0.5, 0.5, 0.5, 0.0, 0.0, 0.5, 0.0, 0.0).$$

where the first four elements refer to $A$ and the last four to $B$. Now, consider an alternative characterization stream obtained by swapping the operands $A$ and $B$. Although the overall transition density remains the same, the resulting bit-wise transition densities are:

$$(A_0, ..., A_3, B_0, ..., B_3) = (0.0, 0.5, 0.0, 0.0, 0.5, 0.5, 0.5, 0.0).$$

Since the multiplication algorithm is likely to require drastically different amounts of computational effort, depending whether the operand is the multiplicand (the value to be added up) or the multiplier (the number of times the multiplicand must be added), it is clear that the power behavior for these two characterization streams will be quite different, although only one of them would be captured in the macromodel.

In the e-HD technique, although two distinct pairs of vectors with same input signal statistics will probably result on
distinct output signal statistics, they are modeled in exactly the same way, which is based on input information only.

Let’s illustrate the impact of such assumption by means of the following example. We performed the characterization of the \textit{Add\_ECLA32} component and monitored each pair \((u, v)\) of successive vectors within the input characterization streams (62 streams with 2000 vectors each). We observed that a collection of 505 pairs had exactly the same input statistics: \(h(u, v) = 15\) and \(s(u, v) = 49\). For such collection, the resulting power estimates lied in the range \([100\mu W, 1000\mu W]\) with a mean value of 482.2\(\mu W\) and a standard deviation of 213.2\(\mu W\). As compared to gate-level reference estimates available for that component, the estimation error lies in the range \([-52\%, 300\%]\). This means that the \(e\)-HD technique may incur high estimation errors because different circuit behaviors cannot be distinguished by the same input signal statistics. If output statistics were included in the model, such distinct behaviors would be better captured.

Since the drawbacks of each method were pointed out qualitatively, we have proper grounds to introduce our multi-model engine in the next section.

4. THE MULTI-MODEL ENGINE

Our multi-model approach consists of four phases, as depicted in Figure 4: (1) individual macromodel creation, (2) individual macromodel evaluation, (3) multi-model creation and (4) multi-model usage. The first three phases are performed only once for a given technology library (during library characterization).

At the left side of the figure, lies the sequence generator, which produces two types of streams: training sets, used during the characterization process (Phase 1) and evaluation sets, used during model robustness evaluation (Phase 2). If a single set was used in Phases 1 and 2, it would capture the intrinsic errors only. To ensure an unbiased macromodeling selection mechanism (Phase 3), training and evaluation sets are generated according to distinct pre-specified parameters.

4.1 Phase 1: Creation of Power Models

Basically, the sequence generator produces as many training sets as the number of components undergoing characterization, although some may be compliant with more than a component and may be reused. The \textit{individual macromodel creation} phase uses the training sets in the characterization of every RTL component from the library for each supported macromodeling technique. As an example, if the library has 100 RTL components and we are using 3 macromodeling techniques, this phase will result in 300 different macromodels (3 for each RTL component).

Although this phase is apparently rather time consuming, given that different macromodels need to be constructed for each design, the time to build the models is actually determined by the technique that requires the higher effort during simulation and model construction. Since all information necessary to build all macromodels is readily available upon simulation completion, the total time will be slightly higher than the time required for the most complex technique.

4.2 Phase 2: Evaluation of Power Models

Once all macromodels are generated for a given RTL component, they are assessed by the evaluation engine. This \textit{individual macromodel evaluation} phase consists in first launching an RTL simulation of the component, which is stimulated by the evaluation set. As a consequence, distinct power estimates are obtained for each macromodel. Then, such estimates are compared to pre-existing gate-level power estimates and an error value is computed. As a result, for each macromodel, a file reporting its robustness on the \(P_{in} \times D_{in}\) space is produced. This information is the input of the multi-model engine, as described next.

4.3 Phase 3: Selection Function Building

Based upon the computed error, a selection function is built to map each point of the input space to the macromodel leading to the least error, as follows.

Let \(M\) be the set of macromodels and let \(e_i\) be the error computed for a given \(i \in M\). The selection function \(\zeta : P_{in} \times D_{in} \rightarrow m\) represents the mapping of a pair \((P_{in}, D_{in})\) to a macromodel \(m\) such that \(e_m = \min\{e_i\}, \forall i \in M\). Since the function \(\zeta\) is highly dependent on a proper choice of input streams, the evaluation set is designed as a huge collection of input streams (approximately 5000 in the current implementation) uniformly distributed over the \(P_{in} \times D_{in}\) space. The function \(\zeta\) is the outcome of the \textit{multi-model creation} phase.

In brief, our multi-model engine associates a distinct function \(\zeta\) to each library component. Given an input stream, its properties \((P_{in}, D_{in})\) are employed to query the function \(\zeta\), which selects the macromodel returning the more accurate power estimate. If a given \((P_{in}, D_{in})\) is not a member of the domain of \(\zeta\), the closest member is chosen according to the least Euclidean distance.

Figures 5(a) and 5(b) display the \(\zeta\)-functions for modules \textit{Add\_ECLA32} and \textit{Mul\_S16}. Each symbol represents the most accurate macromodel selected for a given point of the input...
space. For instance, the symbol \( + \) associated to \( (P_{in}, D_{in}) = (0.8, 0.3) \) in Figure 5(a) indicates that EqTab was the selected model.

Table 1: Results comparing the single-model versus the multi-model approach

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Area ((\mu m^2))</th>
<th>EqTab (\text{min}/\text{max}/\text{avg})</th>
<th>e-HD (\text{min}/\text{max}/\text{avg})</th>
<th>Multi-model (\text{min}/\text{max}/\text{avg})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add_ECLA32</td>
<td>19,351</td>
<td>0.5/34.1/12.9</td>
<td>4.7/71.6/24.9</td>
<td>0.5/33.1/8.5</td>
</tr>
<tr>
<td>MulS16</td>
<td>115,190</td>
<td>2.4/83.4/24.4</td>
<td>5.7/225.3/57.1</td>
<td>0.5/42.4/8.6</td>
</tr>
<tr>
<td>Diffeq</td>
<td>1,573,378</td>
<td>1.7/89.1/26</td>
<td>1.8/433.6/86.9</td>
<td>0.5/53/12.4</td>
</tr>
<tr>
<td>Crossover1</td>
<td>370,343</td>
<td>3.4/64.4/21.2</td>
<td>1.6/184/47.6</td>
<td>0.4/33/9.7</td>
</tr>
<tr>
<td>Crossover2</td>
<td>2,912,320</td>
<td>2.9/65.6/21.7</td>
<td>2.5/269.7/61.3</td>
<td>0.2/33.6/6.7</td>
</tr>
<tr>
<td>Crossover3</td>
<td>3,658,289</td>
<td>2.4/58.2/20.5</td>
<td>2.4/251/61.9</td>
<td>0.3/28.4/8.8</td>
</tr>
</tbody>
</table>

5. EXPERIMENTS

This section compares the estimation accuracy of conventional single-model techniques to the proposed multi-model approach. Reference gate-level estimates were obtained with Synopsys PrimePower.

The adopted benchmark suite consists of six designs, synthesized to a TSMC 0.25\( \mu \)m technology library. The designs’ names are listed in the first column of Table 1, while their areas are shown in the second column. Two are single components from a part library (Add_ECLA32 and MulS16). Four are complex designs, extracted from real-life applications: an implementation of a differential equation algorithm (Diffeq) and implementations of a stereo audio crossover (Crossover1, Crossover2, Crossover3). These complex designs consist of several different components with varying bit-widths, such as adders, multipliers and subtractors. Several estimations were conducted for each design, each with different input streams.

5.1 Common Characterization Set-up

To properly assess the distinct macromodeling techniques, the same vector generation procedure was used during characterization. We adopted the procedure described in [12], which allows input stream generation with high-accurate signal probability, transition density and spatial correlation. As suggested in [12], we set each stream to have 2000 vectors. The adopted input–space range discretization was \((0.00, 0.05, 0.15, \ldots, 0.95, 1.0)\) for both \( P_{in} \) and \( D_{in} \).

5.2 Experimental Results

The results are summarized in Table 1. The first column specifies the distinct design examples, while columns 3 to 5 show the results obtained with the three adopted macromodeling techniques (3DTab, EqTab and e-HD). The last column shows the results for our multi-model approach. Results are expressed as minimum, maximum and average errors, computed as follows:

\[
\varepsilon = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{P_{est}^i - P_{ref}^i}{P_{ref}^i} \right| \times 100\%
\]

where \( n \) is the number of components in the design, \( P_{est}^i \) and \( P_{ref}^i \) are the estimated and reference power values for the \( i \)-th component, respectively.

Note that, the single-model approach may lead to unacceptable errors. For instance, errors higher than 100% were obtained with e-HD. This doesn’t mean that such technique should be discarded beforehand, but show that for some regions in the \( P_{in} \times D_{in} \) space, it is not adequate. However, as we have shown in Section 3, there are regions where the e-HD peformed much better than the others techniques.
Notice that, our maximum estimation error is less than the maximum error of every supported macromodel, except for the smaller design examples (for which we reach the same values as EqTab’s). Most importantly, our approach leads to the least average error for all design examples. To summarize our improvements in a single number, we performed an average on the values in Table 1. As a result, accuracy turns out to be 3 times better on average, as compared to the conventional approaches, while the overall maximum estimation error is divided by 8.

Observe that the adoption of a single macromodel for estimation may lead to unacceptably high errors. In order to correlate the overall system accuracy with the estimation errors of a component, let’s focus on the most complex designs (Diffq, Crossover 1, ..., Crossover 3). Those designs employ MulS16 as a component that dominates the total power consumption. Table 2 shows the estimation errors for distinct instances of MulS16.

<table>
<thead>
<tr>
<th>Inst.</th>
<th>3DTab</th>
<th>EqTab</th>
<th>e-HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>6.14%</td>
<td>1.50%</td>
<td>1.44%</td>
</tr>
<tr>
<td>#2</td>
<td>6.56%</td>
<td>1.17%</td>
<td>9.66%</td>
</tr>
<tr>
<td>#3</td>
<td>2.42%</td>
<td>1.98%</td>
<td>3.69%</td>
</tr>
<tr>
<td>#4</td>
<td>83.93%</td>
<td>21.14%</td>
<td>225.38%</td>
</tr>
<tr>
<td>#5</td>
<td>13.68%</td>
<td>8.21%</td>
<td>1.44%</td>
</tr>
</tbody>
</table>

Table 2: MulS16 robustness example

The large variation in the error value is an evidence of the lack of robustness of a single macromodel. On the one hand, due to such lack of robustness in the estimation of a dominant component, a single macromodel is bound to compromise the overall accuracy. On the other hand, a multi-model approach overcomes this lack of robustness, since the technique leading to the least error is selected (as shown in bold in Table 2).

The overhead imposed to the simulation due to the usage of our multi-model approach for power estimation is minimum. Once all power models have been built and the component from the library appropriately instrumented with its ζ-function, the call to it will only return which model to be used at this call. All remaining overhead is exactly the same as if the single-model approach was used. Thus, as it can be noticed, the robustness improvement obtained by using our approach pays off.

6. CONCLUSIONS

We presented an alternative power estimation approach which exploits multiple macromodels to improve accuracy at the expense of negligible usage overhead. When compared to three distinct conventional macromodeling approaches, accuracy becomes 3 times better on average, while the maximum estimation error is divided by 8.

We believe that multi-model estimation is the key to a wider acceptance of macromodeling in industry, by improving both power simulation performance (as proposed in [2]) and estimation accuracy (as proposed in this paper).

7. ACKNOWLEDGMENTS

We would like to thank prof. Xun Liu, from North Carolina State University, who provided us with his sequence generator.

This work was supported in part by CAPES (0326054), FAPESP (2005/02565-9) and CNPq.

8. REFERENCES