Statistical Design of Macro-models For RT-level Power Evaluation*

Qing Wu, Chihshun Ding, Chengtah Hsieh, Massoud Pedram

Department of Electrical Engineering-Systems
University of Southern California, Los Angeles, CA 90089, USA
Tel: 213-740-4480 Fax: 213-740-7290
e-mail: qingwu@zugros.usc.edu

Abstract— This paper introduces the notion of cycle-accurate macro-models for RT-level power evaluation. These macro-models provide us with the capability to estimate the circuit power dissipation cycle by cycle at RT-level without the need to invoke low level simulations. The statistical framework allows us to compute the error interval for the predicted value from the user specified confidence level. The proposed macro-model generation strategy has been applied to a number of RT-level blocks and detailed results and comparisons are provided.

I. INTRODUCTION

Due to rapid advances in the semiconductor manufacturing technology, the chip density and operating frequency of today's IC's are increasing. Consequently, power dissipation has emerged as a major concern in today's IC's. Low power design requires accurate and efficient estimation tools at various design abstraction levels.

Power estimation at RT-level or higher level is crucial in achieving a short design period. A hierarchical simulation approach to RT-level power estimation is to functionally simulate one circuit and to collect input sequences for each module (major sub-circuit). This information is then passed to various kinds of gate-level or circuit-level simulation programs. The modules are simulated in turn at gate-level or circuit-level using the corresponding input sequences. Finally, the power consumption for all the modules is added together to get the power consumption of the whole circuit. Strictly speaking, this is not an RT-level power estimation methodology because it indeed uses gate-level or circuit-level simulator to do power estimation. Power evaluation is actually done at lower level.

Most RT-level power estimation techniques use capacitance models for circuit modules and activity profiles for data or control signals [1-3]. Such techniques are commonly known as (power) macro-modeling. The simplest form of the macro-model equation is given by:

$$Power = \tfrac{1}{2} V^2 \cdot f \cdot C_{eff} \cdot SW \qquad (1.1)$$

where C_{eff} is the effective capacitance, SW is the mean of the input switching activity, and f is the clock frequency. The Power Factor Approximation (PFA) technique uses an experimentally determined weighting factor, called the power factor, to model the average power consumed by a given module over a range of designs.

More sophisticated macro-model equations can be used to improve the accuracy. Dual Bit Type model, proposed in [2], exploits the fact that switching activities of high order bits depend on the temporal correlation of data while lower order bits behave similarly to white noise data in the data path or memory modules. Thus a module is completely characterized by its capacitance models in the MSB and LSB regions. The break-point between the regions is determined based on the applied signal statistics collected from simulation runs. The Activity-Based Control (ABC) model [4] is proposed to estimate the power consumption of random-logic controllers. All of the above macro-models assume some statistics or properties about the input sequence.

Power macro-modeling formulations in general consist of generating circuit capacitance models for some assumed data statistics or properties. The statistics of input data is gathered during behavioral simulation of the circuit. Power macro-modeling problem defined as follows: Given an input vector sequence of size N, an RT-level circuit with m modules, and assuming N is large enough to capture the typical operation of the circuit, derive a simple function such that the function value of the N vector inputs is as close as possible to the power consumption of the N-vector sequence.

A simple power macro-model equation for the jth module in the circuit may be expressed as:

$$P_j = \tfrac{1}{2} V^2 \cdot f \cdot \sum_{j=1}^{n_j} C_{ij} \cdot SW_{ij} \qquad (1.2)$$

where f is the clock frequency, n_j is the number of inputs for the jth module, C_{ij} is the effective capacitance for input pin i, and SW_{ij} is the switching activity for the ith pin of the jth module. Note that the above equation is only a typical form of macro-model and is not unique. For example, we can include the spatio-temporal correlation coefficients among circuit inputs [5] to improve the power prediction results (this will however significantly increase the number of variables in the macro-model equation and thus the evaluation overhead).

Let P_{jk} denote the power consumption of the jth module at cycle k. We can also write the macro-model equation in a cycle-based form as follows:

$$P_{jk} = \tfrac{1}{2} V^2 \cdot f \cdot \sum_{i=1}^{n_j} C_{ij} \cdot SW_{ijk} \qquad (1.3)$$

where SW_{ijk} is the switching activity (0 or 1) for the ith input of jth module at cycle k. The above equation also illustrate that macro-modeling can be used to estimate the power consumption at each cycle, this ability is critical to our statistical approach. We thus distinguish between *cumulative* macro-models (such as eqn.(1.2)) and *cycle-based* macro-models (such eqn.(1.3)). The total power based on cumulative or cycle-based macro-model can be expressed as:

$$P = \sum_{j=1}^{m} P_j \quad \text{or} \quad P_k = \sum_{j=1}^{m} P_{jk} \qquad (1.4)$$

* This research was supported in part by DARPA under contract number F336125-95-C1627, SRC under contract number 94-DJ-559, and by a grant from Toshiba.

ASP-DAC '97
0-89791-851-7$5.00

where M is the number of modules used in the circuit. To calculate SW_{ij}, behavioral simulation is performed from cycle 1 to cycle N and the mean values of random variates SW_{ij} are tabulated. Let V_{jk} denote the input vector for module j at cycle k, $0 \le k \le N$. A more general macro-model equation for module j at cycle k can be expressed as:

$$P_{jk} = F_j(V_{j,k-1}, V_{j,k}) \qquad (1.5)$$

where F_j could be any function of input vector pairs. Let V_k denote the collection of input vectors, derived from simulation, for m modules at cycle k, $0 \le k \le N$. Then total power equation for cycle k is:

$$P_k = F(V_{k-1}, V_k) \qquad (1.6)$$

where $F = \sum_{j=1}^{m} F_j$.

In the past, average power dissipation has been the primary focus of power estimation techniques and tools. It has however become important to estimate the power distribution of the circuit over a large number of clock cycles. This information is especially useful for determining the circuit reliability, performing dc/ac noise analysis, and choosing appropriate packaging and cooling techniques for IC's. Cycle-based macro-models enable us to predict the circuit power dissipation over time, without the need for low-level simulation.

In general, the three basic criteria for effective macro-model design are:

1. The space and time complexity for collection of parameter values for F and for each evaluation of this function should be small.
2. The accuracy of the macro-model should be high.
3. The error sensitivity of the macro-model to variations in population behavior should be small.

In this paper we propose a statistical design methodology for developing a good cycle-based macro-models for modules (simple or complex cores). The macro-model is built and analyzed based on the theory of regression analysis. A systematic design flow is proposed for model development and verification and two different variable selection methods are discussed. In one approach, detailed information about module (core) structure and functionality is used to derive a *specialized* closed form capacitance equation with a relatively small number of variables. This approach leads to macro-model equations with high accuracy and low evaluation cost. However, it requires detailed knowledge of the module structure and functionality and cannot be fully automated. In the second approach, we start with a *general-purpose* macro-model equation with a large number of variables (for example, all pairwise spatio-temporal correlation coefficients among the module inputs). This technique leads to less accurate macro-models with higher evaluation cost, but the advantage is that it can be fully automated. A variable reduction algorithm is then applied to eliminate as many variables in the general-purpose equation as possible without incurring large errors.

In the paper we discuss the various sources of error due to insufficient training of the macro-model and propose a training set design methodology to make out macro-model universal (error be less sensitivity to variation in population characteristics). Because of our macro-model, which is a multivariate linear regression model, we are able to compute the confidence interval for the estimation of model coefficients. The confidence interval for power prediction of any vector pair can also be evaluated for the purpose of error control. This is a very important and useful feature which is absent from all other power macro-modeling techniques.

II. BACKGROUND

Our goal is to build a cycle-based macro-model which takes a pair of vectors as its inputs and produces a power estimate as its output. The method of linear regression analysis is applied to achieve our purpose.

The statistical relationship between power dissipation and an input vector pair can be defined as,

$$P = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k \qquad (2.3)$$

where P is the power dissipation variable, $\beta_0, \beta_1, \cdots, \beta_k$ are constants called the regression coefficients or parameters of the macro-model, and $X_1, X_2, \cdots, X_k$ are characteristic variables extracted from the input vector pair.

Regression model is a formal means of describing a statistical relation between a set of variables and the characteristic under study. Unlike a functional relation, the statistical relation is not perfect. This means that in general, observations for a regression model do not fall directly on the curve defined by the relationship. There are two essential ingredients of a statistical relation which are expressed by a regression model:

1. Tendency of the dependent variable P to vary with the independent variables $X_1, X_2, \cdots, X_k$ in a systematic fashion,
2. Concentration of points around the surface of statistical relationship.

These two characteristics are embodied in a regression model by postulating that:

1. There is a probability distribution of P for each distinct value of ($X_1, X_2, \cdots, X_k$)
2. The expected values of these probability distributions (distribution means) vary in some systematic fashion with $X_1, X_2, \cdots, X_k$.

Assume that we have been given the equation form of the macro-model and have done simulation (observation) on m randomly sampled vector pairs so that we have obtained m simulation results (observation values) of power consumption. The power linear regression model can be defined as,

$$P_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_k x_{i,k} + \varepsilon_i, \quad i = 1,2,\cdots,m \qquad (2.4)$$

where P_i's are random variates corresponding to observations: ($X_1, X_2, \cdots, X_k$)=($x_{i,1}, x_{i,2}, \cdots, x_{i,k}$); $\beta_0, \beta_1, \cdots, \beta_k$ are the regrssion coefficients; $x_{i,1}, x_{i,2}, \cdots, x_{i,k}$ are known values derived from the input vector pair ($V_{i,1}, V_{i,2}$); and ε_i's are independent random variates representing deviation from the mean value of power. The multivariate regrssion model can be also expressed in matrix form as,

$$\mathbf{P} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \qquad (2.5)$$

Consequently, the random vector $\mathbf{P}$ has an expected value of E[$\mathbf{P}$]=$\mathbf{X}\boldsymbol{\beta}$ and the variance-covariance matrix of $\mathbf{P}$ is COV[$\mathbf{P}$] = $\sigma^2\mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

Because the "real" values of $\boldsymbol{\beta}$ and $\boldsymbol{\varepsilon}$ in the regrssion model are unkown, we apply the method of least squares fit to obtain their estimates **b** and **e**. We denote the vector of estimated regrssion coefficients as,

$$\underset{(k+1)\times 1}{\mathbf{b}} = [b_0, b_1, \cdots, b_k]^{\mathrm{T}} \tag{2.6}$$

The lease squares estimator for the coefficients $\boldsymbol{\beta}$ are:

$$\mathbf{b} = (\mathbf{X}^{\mathrm{T}} \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^{\mathrm{T}} \cdot \mathbf{P} \tag{2.7}$$

It has been proved [6] that the lest square estimator is an unbiased estimator for $\boldsymbol{\beta}$, which means E[**b**]= $\boldsymbol{\beta}$.

The estimated (fitted) power from macro-model is given by the multiplication of input variables and estimated coefficients:

$$\hat{\mathbf{P}} = [\hat{P}_1, \hat{P}_2, \cdots, \hat{P}_m] = \mathbf{Xb} \tag{2.8}$$

and the residual terms are defined as the difference between the fitted power and observed power:

$$\mathbf{e} = [e_1, e_2, \cdots e_m] = \mathbf{P} - \hat{\mathbf{P}} = \mathbf{P} - \mathbf{Xb} \tag{2.9}$$

It is necessary to point out that **b**, **e**, and $\hat{\mathbf{P}}$ are all random variables with certain distributions. We will discuss their statistical properties in Section 3.

Some important statistical properties of regression model [6] are:

error sum of squares: $SSE = \sum_{i=1}^{m} e_i^{\,2}$

error mean squares: $MSE = SSE/(m-k-1)$

regrssion sum of squares: $SSR = \sum_{i=1}^{m} (\hat{P}_i - \overline{P})^2$

regrssion mean squares: $MSR = SSR/k$

coefficient of multiple correlation: $R = \sqrt{SSR/(SSR+SSE)}$

III. Building the Regrssion Model

Our workflow of building a good cycle-based macro-model is shown in Figure 3.1

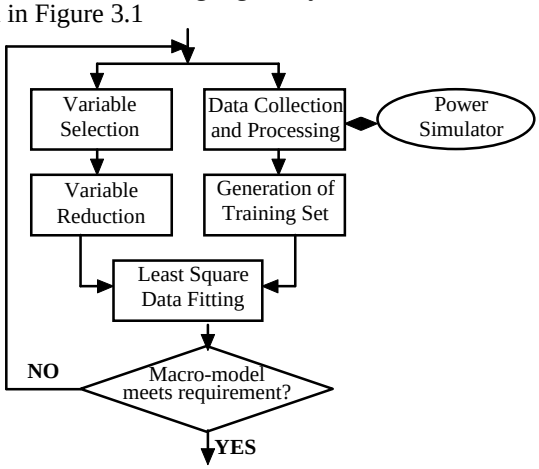
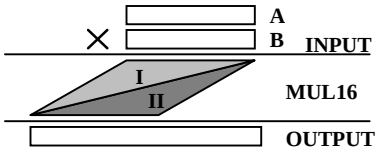


Fig.3.1 The workflow of developing a macro-model

3.1 Variabel selection

Variable selection is the first important step for building a good macro-model.

If detailed information about the module (core) structure and functionality is provide. Then this information can be used to derive a macro-model form with a relatively small number of variables, yet high accuracy. We call this a *specialized* macro-model. As an example, consider variable selection for MUL16, which has the structure shown below:



The structure of MUL16 is basically a plane of 256 AND gates integrated with 1-bit full adders. The power dissiation of these adders is determined by the switching activity on their inputs. We divided the plane into two symmetric parts as shown above. Part I consists of 120 AND gates and 1-bit adders while part II consists of 136 AND gates and 1-bit adders. Let us define transition type as either 00, 01, 10, or 11 transition. We use four variables X_{ij} (i,j=0,1) to represent the number of transitions of type ij at the ouputs of the AND gates in part I. Similarly, we use four variables Y_{kl} (k,l=0,1) to represent the number of transitions of type kl at the outputs of the AND gates in part II. We then introduce second order terms $X_{ij}X_{kl}$, $X_{ij}Y_{kl}$, $Y_{ij}Y_{kl}$ for a total of 10+16+10=36 variables. In total, the number of variables in the specialized MUL16 macro-model becomes 44. Note that all variable values can be derived from knowledge of the input patterns which are applied to MUL16, that is, no low-level simulation is required.

The procedure for generating a specialized macro-model cannot be fully automated, because it requires detailed analysis of the structure and functionality of a module. In our general purpose macro-modeling approach, the original variable set contains the variables that reflect the transition situation of each input and the pairwise spatial correlation between every pair of them. A variable reduction algorithm is then applied to choose a "best" subset from the original variables as the final selected variable for the macro-model as detailed next.

3.2 Variable reduction

Number of variables in the initial macro-model can be in the hundreds. The *forward regression procedure* [6] is the most suitable automatic search method for a regression model with this many variables. The search method develops a sequence of regression models, at each step adding or deleting one X variable. The criterion used for adding or deleting variables is the F* statistics [6] in regression analysis. The algorithm (which assumes a linear macro-model equation form similar to Eqn.(2.3)) is described below:

Input : Given are a set of candidate variables $\{X_1, X_2, \cdots, X_N\}$, a training set, a low threshold t_0, a high threshold t_1. S is a set of selected variables.

Step 0 : Set S=Φ and C = $\{X_1, X_2, \cdots, X_N\}$.

Step 1 (start) : Fit a one-variable linear regression model for each of the X variables. The F* statistic for each model is obtained by:

$$F_i^* = \frac{MSR(X_i)}{MSE(X_i)}, \qquad i = 1,2,\cdots,N \tag{3.1}$$

Assume that X_j is the variable with the maximum F^* value. If $F_j^* \geq t_1$ then move X_j from C to S and denote it as X_1^*. Otherwise, no macro-model can be found for the given t_1 value (t_1 must be reduced). The algorithm terminates.

Step 2 (add variable) : Assume $S = \{ X_1^*, X_2^*, \cdots, X_a^* \}$, for each X_i remaining in C, fit the regression model with a+1 variables $X_1^*, X_2^*, \cdots, X_a^*$ and X_i . For each of them, the partial F test statistics is:

$$F_i^* = \frac{MSR(X_i | X_1^*, X_2^*, \cdots, X_a^*)}{MSE(X_i, X_1^*, X_2^*, \cdots, X_a^*)} = (\frac{b_i}{s\{b_i\}})^2 \qquad (3.2)$$

where b_i is the estimated value of β_i coefficient and $s\{b_i\}$ is the standard deviation of b_i. Let X_j be the variable with the maximum F_i^* value. If $F_j^* \geq t_1$ then move X_j form C to S and denote it as X_{a+1}^*, increase a by 1, and go to Step 3; Otherwise the algorithm terminates.

Step 3(delete variable) : Assume S=$\{ X_1^*, X_2^*, \cdots, X_a^* \}$, and X_a^* is the latest variable added in Step 2. Compute the partial F test statistics for all other variables in S:

$$F_i^* = \frac{MSR(X_i^* | X_1^*, X_2^*, \cdots, X_{i-1}^*, X_{i+1}^*, \cdots, X_a^*)}{MSE(X_i^*, X_1^*, X_2^*, \cdots, X_a^*)} = (\frac{b_i}{s\{b_i\}})^2, \qquad i = 1,2,\cdots,a-1 \qquad (3.3)$$

Let X_j^* be the variable with minimum F^* value. If $F_j^* < t_0$ then remove X_j^* form S.

Step 4 : Repeat Steps 2 and 3 until algorithm terminates in Step 2 or there is no variable in the candidate set C.

With user defined threshold t_0 and t_1, the above algorithm will find the "best" variables for the macro-model from the candidate set. The number of "best" variables retained in the model is controlled by assigning appropriate threshold values.

3.3 Training set design

Definition. Population is the set of all possible input vector pairs applied to a module.

Definition. Training set is a representative subset of the whole population which is used to estimate coefficients of the macro-model.

The general requirement for generating the training set for a macro-model is that it should create the ranges of all possible values of independent variables X_i and dependent variable P in the original population . When either of these ranges is not sufficiently covered by the training set, we say that the macro-model is not well trained or, more precisely, it is insufficiently trained. According to the source of insufficiency (range of X_i's versus range of P), insufficiently trained macro-models can be classified into type I versus type II.

When applying the macro-model to new subsets of the population (i.e., subsets other than the training set), the insufficiently trained macro-model of type I will, in most cases, results in large errors. Normally this problem can be solved by doing more experiments and collecting more fitting data from the available population. Table 3.1 shows the error values caused by the C1908 macro-model (66 variables) using training sets of different sizes. The units in the training sets are randomly sampled from the population. Using the training sets of different sizes, macro-models with different coefficients were obtained and applied to estimate the power dissipation for whole population.

In the Table, the average error and sum error is computed by:

$$\textbf{EAP}(\text{Error in Average Power}) = \frac{\left| \sum_{i=1}^{N} \hat{P}_i - \sum_{i=1}^{N} P_i \right|}{\sum_{i=1}^{N} P_i} \qquad (3.4)$$

$$\textbf{ECP}(\text{Error in Cycle Power}) = \frac{1}{N} \sum_{i=1}^{N} \frac{\left| \hat{P}_i - P_i \right|}{P_i}$$

where N is the size of the population, $\hat{P}_i$ is the estimated power for unit i, and P_i is the corresponding "actual" power value.

TABLE 3.1
THE ERROR CAUSED BY THE MACRO-MODEL TRAINED BY TRAINING SETS OF DIFFERENT SIZES

Training Set Size	ECP	EAP
100	33.54%	4.90%
200	19.74%	1.07%
500	14.90%	1.29%

Results show that, when size of the training set is too small, the full range of range of values of variables in the macro-model equation is not exercised sufficiently, resulting in larger errors. However, after the size of training set surpasses the lower bound for efficient training, the accuracy of the macro-model can be hardly improved by using more training units.

The magnitude of the error caused by insufficiently trained macro-model of type II depends on the difference in the characteristic under study (for example, power range) between the new sequence and the training set. This problem, which is called the population-sensitive error problem, is more difficult to overcome and can not be completely avoided.

Table 3.2 shows experimental results of the population-sensitive error problem. In the experiment, we used three different training sets and their union to train and get four MUL16 macro-models with different coefficients. Then we applied each one of these macro-models to all three sets and their union separately to evaluate the errors. The three training sets {A, B, C} correspond to input sequences going through a MUL16 in three different applications. Training set A is digitalized music waves. Training set B is random white noise input. Training set C is obtained from a filtering application in which one of the data operands is fixed. Because the sizes of sets A and B are much larger than set C, the union set is dominated by sets A and B.

Because sets A and B have similar power characteristics, the macro-model trained by one of them has good accuracy when

TABLE 3.2
EXPERIMENTAL RESULTS FOR POPULATION-SENSITIVE ERROR PROBLEM

Training Set	Size	Range (μW)	A		B		C		A+B+C	
			ECP (%)	EAP (%)	ECP (%)	EAP (%)	ECP (%)	EAP (%)	ECP (%)	EAP (%)
A	3000	[14,122]	10	0*	11	6.4	-**	67	-**	6.7
B	3500	[16,143]	13	1.2	11	0*	-**	97	-**	3.0
C	630	[0,57]	870	660	390	205	-**	0*	-**	378
A+B+C	7130	[0,143]	12	2.4	11	1.5	-**	3.4	-**	0*

* When the regression macro-model is applied to its training set, the error of average power is always zero

** Eqn.(3.4) is not applicable for average error computation because there are units in set C which have zero power consumption

applying to another. But set C has quite different characteristics from sets A and B. As a result, the macro-model trained by set C cause large error on sets A, B and the union set. The macro-model trained by sets A or B is not applicable to set C either. However, the macro-model trained by the union set, which covers all the power range of set A, B, C, has very good accuracy on all the sets.

It is desirable to have a macro-model which remains accurate regardless of the specific subset of the population it may encounter in practice. One way of doing this is to build different macro-models for different sub-populations. In practice, we will first analyze the characteristics of the input data applied to the module, then apply the appropriate macro-model. This methodology is similar to the population-partition method in building specialized macro-models. However in most cases, the population characteristics varies widely and it is not possible to derive some well-behaved population partitioning scheme. Even in the same application, different instances of a module may encounter very different population characteristics based on the circuit context in which they are embedded. As a result, designers prefer to have a single static macro-model that can be used in all kinds of applications, in other words, a universal macro-model.

Generation of the training set is also an important step to design a good universal macro-model. In this paper, we generate the training sequence through population stratification and random sampling as described next.

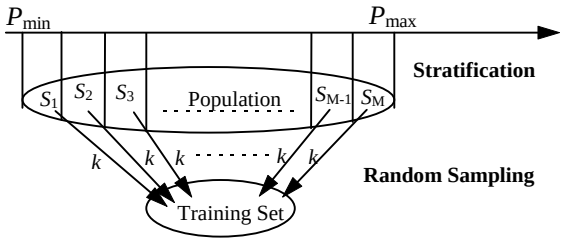


Fig.3.3 Generating the training set using stratified random sampling

In the first step, data is collected from all applications in which the module is instantiated (for example, through architectural or behavioral simulation of the system which contains the module) or it is generated by automatic sequence generation techniques [7] (which take signal and/or transition probability and generate short sequences that satisfy the specified behavior). Assume that this data covers the whole range of the power consumed by the module. Let P_{min} and P_{max} denote the minimum and maximum power among all the units. We divide equally the region between P_{min} and P_{max} into M sub-regions, thus forming M strata. According to its power consumption, every unit may fall in exactly one of these strata. Next, we randomly select k units from each stratum to put into the training set. Finally, we get the training set of size $m=M*k$. By using the stratified random sampling technique [8], the size of the training set is largely reduced while the property of the population is captured by the training set. In this way, we ensure that the macro-model will be sufficiently trained to keep type II error in check.

3.4 Inference about prediction of new observations

Information about the estimation error is the key factor to improve the accuracy. When we apply the macro-model to predict the power of an input vector pair, we like to know not only the estimated power value, but also the estimation error. One major advantage of using regression analysis as described above is that the regression macro-model can predict the power consumption for an input pair and give confidence interval of the prediction for a given confidence level as detailed next.

Values of regression variables, X_{obs}, are extracted from each input vector pair (V_{i-1}, V_i). These variables are then plugged into the macro-model equation to yield the power estimation result $\hat{P}_{obs}$ for the vector pair. At this point, we do not know the actual value of the observation, that is, the "real" simulation result for the vector pair. What we are able to do however is to derive a confidence level for the unknown observation.

Given a confidence level 1-α, the corresponding *confidence interval* is the interval $[u_1, u_2]$ such that the probability that the actual power value lies inside this interval is 1-α.

Given a confidence level 1-α, the confidence interval of the observation P_{obs} is given by:

$$[\hat{P}_{obs} - t(1-\alpha/2; m-k-1)\cdot s[P_{obs}], \quad \hat{P}_{obs} + t(1-\alpha/2; m-k-1)\cdot s[P_{obs}]] \tag{3.11}$$

where $t(1-\alpha/2;m-k-1)$ is the $(1-\alpha/2)\times 100$ percentile point of the t distribution with degree of freedom of $(m-k-1)$ and $s[P_{obs}]$ is standard deviation of the new observation which is given by:

$$s[P_{obs}] = \sqrt{MSE \cdot (1 + X_{obs}^{T}(\mathbf{X}^{T}\mathbf{X})^{-1} X_{obs})} \tag{3.12}$$

Note that **X** and MSE refer to the training set matrix values and mean square error of the training set as defined in Section II. In simple terms, the probability that the absolute value of the difference between $\hat{P}_{obs}$ and P_{obs} exceeds $t(1-\alpha/2;m-k-1)\cdot s[P_{obs}]$, is α.

Table 3.3 gives the experimental results of error computation for C1908 general purpose macro-model at a confidence level of 95%. For example row 1 tells us that for the vector pair (selected randomly), the estimated power value was 1.302*mW*, while the actual power (measured by PowerMill [10]) was 1.537*mW*. Furthermore, the confidence interval for a 95% confidence level was calculated to be [0.74, 1.87]*mW*. Since the actual value lies within the confidence interval, we have a correct prediction. This is not true for the second vector pair since the actual value is outside the confidence interval. This is statistically possible since we can ensure that only 95% of the time, the actual value will be within the

confidence interval. Note that when the confidence level is decreased, the confidence interval also shrinks, and vice versa.

TABLE 3.3
EXPERIMENTAL RESULTS OF ERROR CONFIDENCE INTERVAL PREDICTION

Vector Pair	Estimated Power (mW)	Confidence Interval (mW)	Actual Power (mW)	Correct Prediction
1	1.302	[0.74, 1.87]	1.537	YES
2	1.323	[0.78, 1.87]	1.944	NO
3	1.329	[0.76, 1.90]	1.213	YES
4	1.274	[0.70, 1.85]	1.499	YES
5	1.765	[1.23, 2.30]	1.676	YES
6	3.095	[2.56, 3.63]	2.808	YES
7	1.994	[1.46, 2.53]	2.325	YES

Since the fitted power value follows normal distribution, from our observation on the experimental results for more than 10 circuits. the average error is approximately 1/4th of the confidence interval.

Introducing the notion of the confidence interval into high-level power estimation provides us the means to control the error and improve the accuracy of the estimates as shown below.

Predefine a confidence level (1-α) (for example, 95%) and a tolerance limit for error (for example, 10%). For each clock cycle, use the macro-model to estimate the power consumption of the module in that cycle. At the same time, the confidence interval is computed by eqn.(3.11). According to the estimated power and the error tolerance limit, we can also compute the tolerable region for error (we call it error tolerance interval). If the confidence interval totally falls within the error tolerance interval, then the error is tolerable at this confidence level and we accept the macro-model estimate. Otherwise, the error is not tolerable at this confidence level and we must use a more complex and more accurate macro-model to estimate the power.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

Some experimental results are shown in Table 4.1.

TABLE 4.1
EXPERIMENTAL* RESULTS FOR SOME SPICIALIZED AND GENERAL PURPOSE MACRO-MODEL

Module	Type	No. of Var's	ECP (%)	EAP (%)
MUL16	Specialized	44	10.6	4.3
C6288	Specialized	44	7.9	3.1
ADD16	General	64	8.0	1.0
MUL4	General	80	9.4	1.2
C1355	General	82	13.3	10.9
C1908	General	66	15.4	2.3
C3540	General	78	15.5	5.2

* In experiments, the training sets are subsets of the whole populations for different modules. The error in cycle power and error in average power are computed on applying the macro-model to whole population.

It can be seen that the average cycle-based error is 11.4% while the average total power error is 4%.

When the user only wants to estimate the average power dissipation of a module, a cumulative macro-model is applied. Transforming the cycle-base macro-model to cumulative macro-model for estimating average power is very simple. Assume that the cycle-based macro-model is:

$$P = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k \qquad (4.1)$$

Then, the cumulative macro-model for average power estimation is:

$$\overline{P} = \beta_0 + \beta_1 \mathbf{E}[X_1] + \beta_2 \mathbf{E}[X_2] + \cdots + \beta_k \mathbf{E}[X_k] \qquad (4.2)$$

If $X_1, X_2, \cdots, X_k$ are all 0-1 variables, the cumulative macro-model becomes:

$$\overline{P} = \beta_0 + \beta_1 \mathrm{Prob}[X_1] + \cdots + \beta_k \mathrm{Prob}[X_k] \qquad (4.3)$$

V. CONCLUSION

In the paper, we intrduced the notion of cycle-based macro-models for RT-level power estimation. In this way we are able to esitmate not only the average power consumption at RT-level, but also the power distribution over all the cycles that are simulated. The macro-model is built on the basis of regrssion analysis. Two variable selection strategies were discussed: specialized and general purpose. The number of variables can be reduced using statistical tests. The statistical methodology enables us to not only predict the power values at RT-level without invoking low level simulators, but also compute the error and confidence level for our prediction. The technique was applied to generate macro-models for various RT-level cores and achieved good accruacy.

REFERENCES

[1] S. Powell and P. Chau, "Esitmating power dissipation of VLSI signal processing chips: The PFA techniques", *Proceeedings of IEEE Workshop on VLSI Signal Processing IV*, vol. IV, pp.250-259, 1990.

[2] P.Landman and J. Rabaey, "Power esitmation for high-level synthesis", *Proceedings of IEEE European Design Automation Conference*, pp.361-366, Feb, 1993.

[3] D. Liu and C. Svensoon, "Power consumption estimation in CMOS VLSI chips", *IEEE Journal of Solid State Circuits*, vol. 29, pp.663-670, Jun. 1994.

[4] J. Rabaey and P. Landman, "Activity-sensitive architectural power analysis for the control path", *Proceedings of International Symposiom of Low Power-Design*, pp.93-98, Apr. 1995.

[5] R. Marculescu, D. Marculescu, and M. Pedram, "Logic level power estimation considering spatiotemporal correlations", *Proceedings of the IEEE International Conference on Computer Aided Design*, pp.294-299, Nov. 1994.

[6] J. Neter, W. Wasseman, and M. H. Kutner, *Applied Linear Regrssion Models*, Second Edition, Richard D. Irwin, Inc, 1989.

[7] D. Marculescu, R. Marculescu, and M. Pedram, "Stochastic sequential machine synthesis targeting constrained sequence generation", *Proceedings of the Design Automation Conference*, pp.696-701, Jun. 1996.

[8] C. Ding, C. Hsieh, Q. Wu, and M. Pedram, "Stratified Random Sampling for Power Estimation", to appear in *Proceedings of the International Conference on Computer Aided Design*, 1996.

[9] A. T. Craig and R. V. Hogg, *Introduction to Mathematical Statistics*, Fourth Edition, Macmillan Publish, 1978.

[10] PowerMill User manual, Release 3.1, EPIC Design Technology, 1994