Statistical Design of Macro-models for RT-level Power Evaluation

Qing Wu and Massoud Pedram

CENG 96-24

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213) 740-4458

June 1996

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

Abstract

This paper introduces the notion of cycle-based macro-models for RT-level power evaluation. They provide us the capability to estimate power dissipation cycle by cycle at RT-level. They give us not only the average power but also the distribution of the power dissipation over the all cycles being simulated (e.g., the peak power). The mathematical foundations of this macro-model technique are statistical sampling for the training set design and regression analysis combined with appropriate statistical tests for the macro-model variable selection and coefficient calculation. The statistical framework enables us to both predict the power values without the need to invoke low level simulations and to compute the error and confidence level for the predicted value (with respect to "actual" power dissipation). The proposed macro-model generation strategy has been applied to a number of RT-level blocks (adders, multipliers, etc.) and detailed results and comparisons are provided, confirming that automatic generation of power macro-models for various RT-level cores is feasible and the estimation reaches the acceptable accuracy at RT-level.

Section 1. Introduction and motivation

Due to rapid advances in the semiconductor manufacturing technology, the chip density and operating frequency of today's ICs are increasing. Consequently, power dissipation has emerged as a major concern in today's IC's. Low power dissipation results in lower packaging cost and higher circuits reliability. Portable electronic devices using batteries are another important driving force for low power design. Low power design requires accurate and efficient power estimation tools at various design abstraction levels.

Numerous power estimation methods have been developed at gate-level or circuit-level estimation to achieve two goals: low computational cost and high accuracy. Although real-delay gate simulator and circuit-level simulator can provide accurate results, we cannot afford the time it takes them to simulate tens of thousands of vectors. Probabilistic techniques were developed to increase the estimation speed. But the methods rely on simplified circuit models and become inaccurate for real-delay power estimation. Recently developed statistical techniques use survey sampling methods to get a representative subset of the original sequence. The size of the subset is much smaller than the original sequence. By simulating the subset using accurate simulator, an accurate average power consumption can be obtained within a short time.

Power optimization at RT-level or higher level is crucial to achieve a short design period. In hierarchical simulation techniques, circuit is simulated at RT-level functionally. The input sequence for each module in the circuit is then collected and passed to various kinds

of gate-level or circuit-level simulators. 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, hierarchical simulation 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 [1] 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 the 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 is 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 could be expressed as:

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

where f is the clock frequency, n_j is the number of inputs for the jth module, C_i 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 illustrates 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 *one-shot* macro-models (such as eqn.(1.2)) and *cycle-based* macro-models (such as eqn.(1.3)). The total power based on one-shot or cycle-based macro-models can be expressed as:

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

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_{ih} 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 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. 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 costs, 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 our 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.

The cycle-based macro-model is a Section of our power estimation framework shown in Figure 1.1.

The estimation framework is a multi-level co-simulation environment which is constructed by RT-level, gate-level and circuit-level simulation. Firstly the circuit is simulated functionally using an RT-level functional simulator. The input information for all the modules in the circuit is collected. Input sequences for modules are passed to macro-models immediately to evaluate power consumption without the need to invoke low-level simulators (i.e., circuit or gate level simulators). Statistical sampling techniques also take the input sequences for each macro-model and output a subset to the simulators to do partial simulation for power. Our cycle-based macro-model can be used in

conjunction with statistical sampling techniques to provide a highly effective and accurate RT-level power estimation methodology.

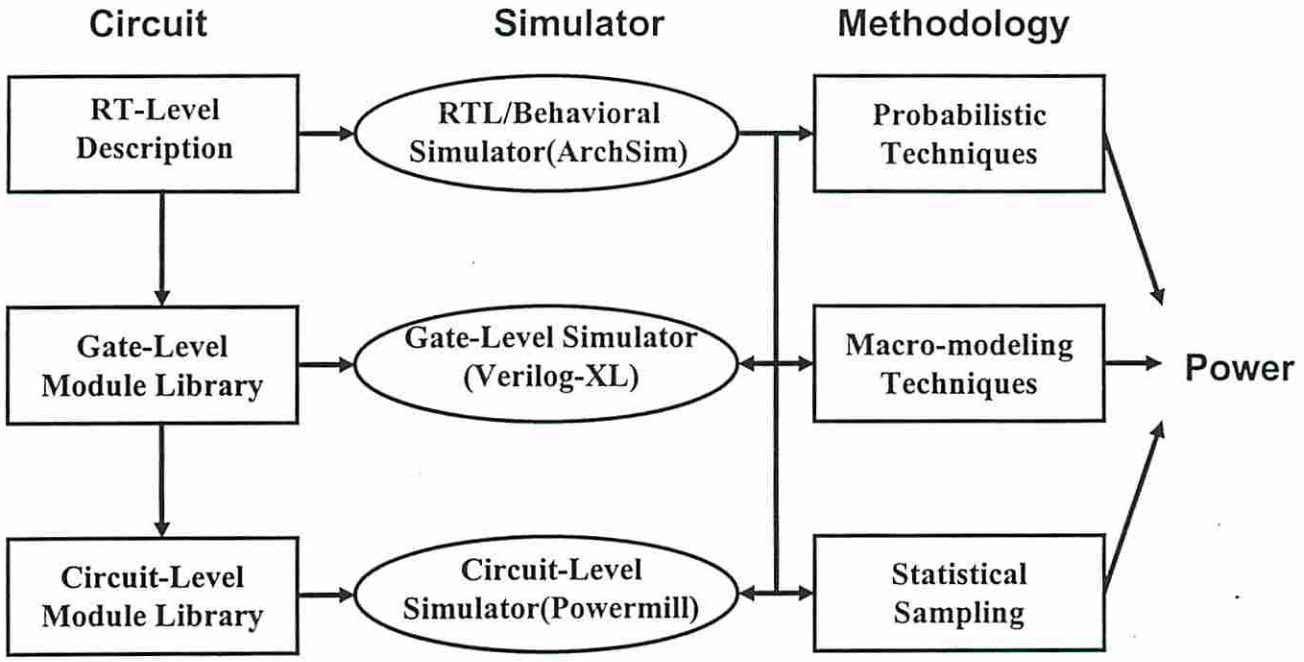


Figure 1.1 Power estimation framework

Section 2. Definitions and basic knowledge of regression analysis

Power consumption of a combinational module is solely determined by the transition activities of its primary inputs. Specifically, from the time when primary input transitions occur until all internal nodes and primary outputs become stable, the energy consumed by a particular module can be defined as a function of two consecutive input vectors,

$$Energy = f(V, V') \tag{2.1}$$

where $V = (v_1, v_2, \cdots, v_n)$ and $V' = (v_1', v_2', \cdots, v_n')$ are two consecutive vectors applied to the n primary inputs. In synchronous circuits, input vector application is synchronized by a clock. Hence, the power consumption by the module in clock cycle i can be defined as,

$$P_i = g(V_{i-1}, V_i) = \frac{1}{T_{CLOCK}} f(V_{i-1}, V_i) \tag{2.2}$$

where V_{i-1} and V_i are input vectors at clock cycles i-1 and i, respectively. We call any two consecutive vectors in the input sequence a vector pair.

In general, function g is a complicated and highly non-linear function of input vector pair. It is determined by the detailed structure of the module and its actual implementation in a given technology. In some sense, g can be viewed as gate-level or circuit-level simulator which is evaluating power consumption of the module in a cycle. Obviously, we cannot get a closed form of g which is exact.

Our goal is to build a cycle-based macro-model which takes a vector pair as its inputs and produce 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,

$$P = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k \tag{2.3}$$

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. A scattering 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 give 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,

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

where:

P_i's are random variates corresponding to observations : $(X_1, X_2, \ldots, X_k) = (x_{i,1}, x_{i,2}, \ldots, x_{i,k})$

$\beta_0, \beta_1, \cdots, \beta_k$ are the regression coefficients

$x_{i,1}, x_{i,2}, \cdots, x_{i,k}$ are known values derived from the input vector pair $(V_{i,1}, V_{i,2})$
ε_i's are independent random variates representing deviation from the mean value of power.

The multivariate regression model can also be expressed in matrix terms as,

$$\mathbf{P} = \mathbf{X}\beta + \varepsilon \tag{2.5}$$

where:

$$\underset{n\times 1}{\mathbf{P}} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_m \end{bmatrix}, \quad \underset{n\times(k+1)}{\mathbf{X}} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,k} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & x_{m,2} & \cdots & x_{m,k} \end{bmatrix}, \quad \underset{(k+1)\times 1}{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}, \quad \underset{n\times 1}{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{bmatrix}$$

Consequently, the random vector $\mathbf{P}$ has an expected value of $\mathbf{E}[\mathbf{P}] = \mathbf{X}\beta$ and the variance-covariance matrix of $\mathbf{P}$ is $\mathbf{COV}[\mathbf{P}] = \sigma^2\mathbf{I}$, where $\mathbf{I}$ is the identity matrix.
Because the "real" values of β and ε in the regression model are unknown, we apply the method of least squares fit to obtain their estimates $\mathbf{b}$ and $\mathbf{e}$. We denote the vector of estimated regression coefficients as,

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

The least squares estimator for the coefficients β 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 least square estimator is an unbiased estimator for β, which means $\mathbf{E}[\mathbf{b}] = \beta$.

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

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

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

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

It's necessary to point out that **b**, **e**, and $\hat{\mathbf{P}}$ are all random variables of certain distribution. 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: $MSE = SSE/(m-k-1)$

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

regression mean squares: $MSR = SSR/k$

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

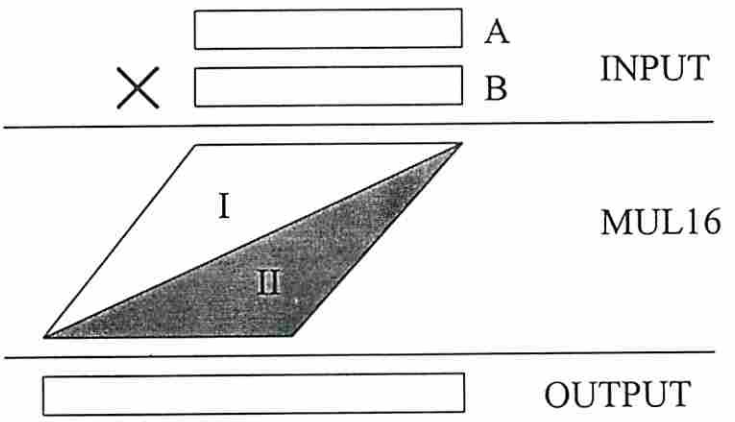
Section 3. Building the regression model

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

3.1 Variable selection

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

When detailed information about the module (core) structure and functionality is provided. They can be used to derive the macro-model form with relatively small number of variables and high accuracy. We call it *specialized* macro-model. As an example, when we are selecting variables 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 dissipation of those adders is determined by the transition situation on their inputs. We divided the plane into two symmetric part as shown above. Eight basic variables are selected representing the number of four transition type (00, 01, 10, 11) on the outputs of 120 and 136 AND gates in part I and part II, respectively. To capture the

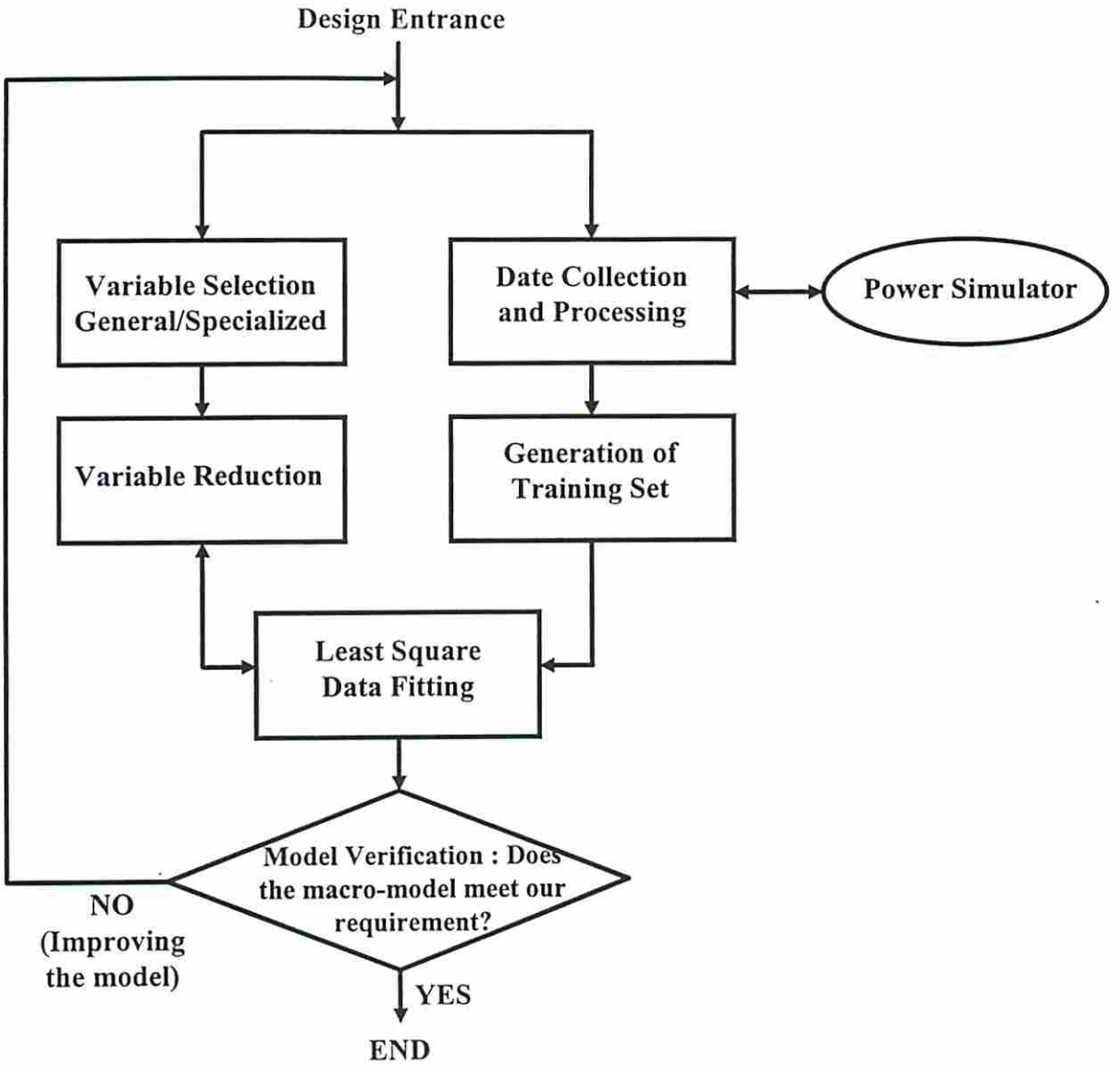


Figure 3.1 The workflow of developing a macro-model

nonlinear relationship between transition number and power, Second order terms of basic variables are added to the macro-model equation. In total, the number of variables in a specialized MUL16 macro-model is 44.

However, the procedure for generating a specialized macro-model cannot be fully automated because it requires intellectual analysis on the structure and functionality of a module. While the variables of a *general purpose* macro-model are input transition situation and the spatio-temporal correlations between inputs. The procedure of generating a general purpose macro-model can be fully automated without knowing any information about the module.

In our approach, the original variable set contains the variables that, reflecting the transition situation of each input, and reflecting the pairwise spatial correlation between every two of them. The variable reduction algorithm is then applied to choose a "best" subset (of size below 80) from the original variables as the final selected variable for the macro-model.

3.2 Variable reduction

There are a number of algorithms to do variable reduction. The *all -possible-regression selection procedure* [6] calls for a consideration of all possible regression models involving the potential X variables and identifying a few "good" subsets according to some criterion. But the exponential relation between spatial/time complexity and the number of variables makes the procedure inappropriate for a regression model with a large number of variables. The "best" subsets algorithms [6] are developed to investigate only a small fraction of all possible regression models. However when the number of variables is larger than 60, they also require excessive computer time.

In our application, the number of variables in the original macro-model could be in the hundreds. The *forward stepwise 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 is described below:

Input : Given are a set of candidate variables $\{X_1, X_2, \ldots, X_N\}$, a training set, a low threshold t_0, a high threshold t_1, and an adjustment step size Δ. S is a set of selected variables.

Step 0 : Set $S = \phi$ and $C = \{X_1, \ldots X_N\}$.

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

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

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^*, go to Step 2. Otherwise decrease t_1 by Δ and redo Step 1.

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

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

where b_i is the estimated coefficient for variable X_i and $s\{b_i\}$ is the standard deviation of b_i. Let X_j be 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_{a+1}^*, increase a by 1, and go to Step 3. Otherwise the algorithm terminates.

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

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

Let X_j be the variable with minimum F^* value. If $F_j^* < t_0$ then remove X_j from S, else retain it.

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 thresholds t_0 and t_1, the above algorithm will help 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, 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 larger 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 of whole population.

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

$$\text{average error} = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\hat{P}_i - P_i\right|}{P_i} \quad \text{and, sum error} = \frac{\left|\sum_{i=1}^{N}\hat{P}_i - \sum_{i=1}^{N}P_i\right|}{\sum_{i=1}^{N}P_i} \tag{3.4}$$

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

Table 3.1 The error caused by the macro-model trained by training sets of different sizes

Size of Training Set	average error	sum error
100	33.54%	4.90%
200	19.74%	1.07%
500	14.90%	1.29%

The results show that, when the size of the training set is too small, the range of value of variables in the macro-model equation is not exercised sufficiently. It results in larger error. However, after the size of training set passes the lower bound of efficient training, the accuracy of the macro-model can hardly be 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. Indeed it is a major problem that no macro-modeling technique can completely avoid. The problem is depicted graphically in Figure 3.2.

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 set A and set B are much larger than set C, the union set is dominated by sets A and B.

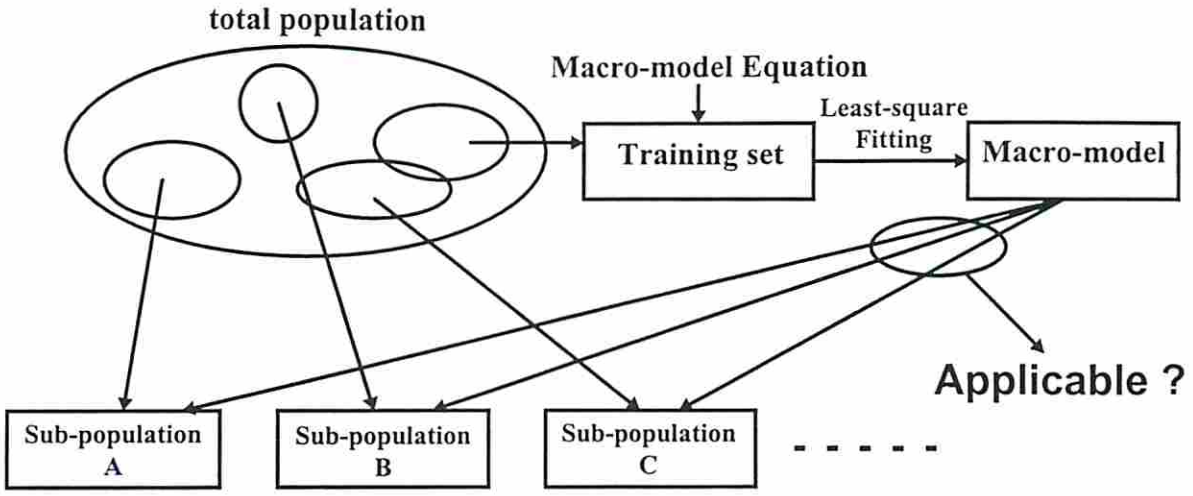


Figure 3.2 The population-sensitive error problem

Table 3.2 Experimental results for population-sensitive error problem

Training Set	Size	Power Range (μW)	A		B		C		A+B+C	
			Ave. Err	Sum. Err	Ave. Err	Sum. Err	Ave. Err	Sum. Err	Ave. Err	Sum. Err.
A	3000	[13.5, 122]	10.7%	0*	11.0%	6.43%	-**	67.4%	-**	6.7%
B	3500	[16.1, 143]	12.8%	1.21%	10.5%	0*	-**	97.2%	-**	3.0%
C	630	[0, 56.7]	870%	660%	390%	205%	-**	0*	-**	378%
A+B+C	7130	[0, 143]	11.7%	2.39%	10.8%	1.45%	-**	3.35%	-**	0*

* When the regression macro-model is applied to its training set, the sum error is always zero.
** Equation(3.4) is not applicable for average error computation because there are units in set C with zero power dissipation.

Because set A and set B have similar power characteristics, the macro-model trained by one of them has good accuracy when applying to another. But set C has quite different characteristics from set A and B. As a result, the macro-model trained by set C caused large error on set A, B and the union set which is dominated by A and B. The macro-model trained by set 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 encounter in practice. One way of doing this is to build different macro-models for different subpopulations. 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 define some well-behaved population partition. 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 having 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 stage to design a good universal macro-model. In this paper, we generate the training sequence through population stratification and random sampling.

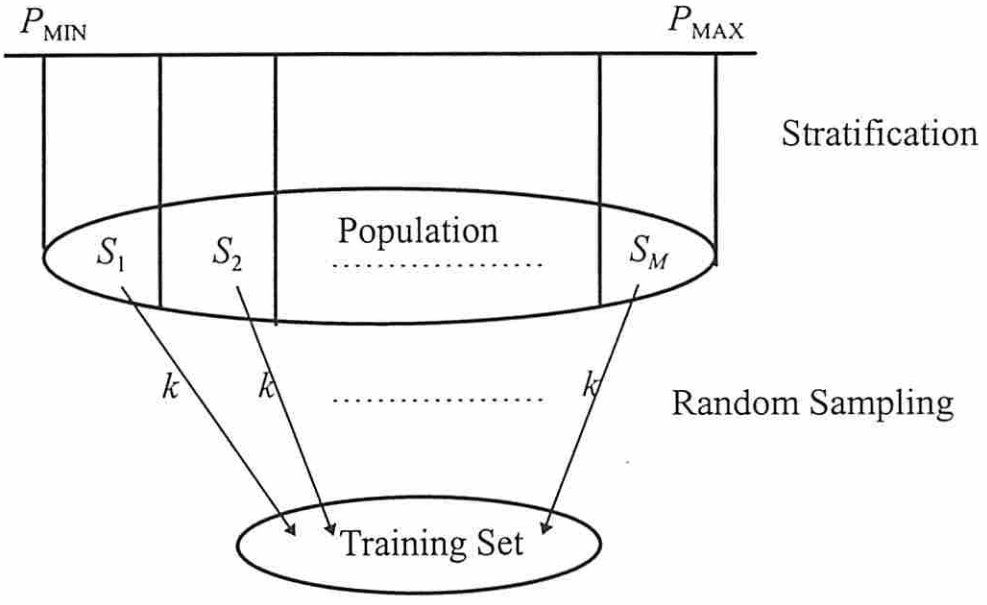


Figure 3.3 Generating the training set using stratified random sampling

In the first step, data is collected from all applications in which the module is instantiated (for example, during 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 satisfies 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 the region between P_{MIN} and P_{MAX} equally into M sub-regions. Thus we form M strata. According to its power consumption, every unit may fall in one and only one of these strata. Next, we randomly sample k units from each stratum to put into the training set. Finally, we get the training set of size $m=M\times k$. By using 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 counter (if not eliminate) type II error.

3.4 Inferences and verification

As we defined in Section 2, a power macro-model should be viewed as a statistical relation between selected variables and power consumption. The least-square estimated coefficients b_i, the fitted power values $\hat{P}_i$, and the error terms e_i, are all random variables with certain distribution functions. The statistical inferences on these variables is a formal way of inspecting the quality of the macro-model.

3.4.1 Inferences about the estimated coefficients b_i

When defining the linear regression model, we assume the error terms ε_i in eqn.(2.4) follow the normal distribution with mean 0 and variance σ^2, that is, $N(0,\sigma^2)$. Consequently, P_i's follow the normal distribution. According to eqn.(2.7), b_i's are the linear combination of P_i's, thus b_i's also follows the normal distribution. In practice, the error terms could not be strictly normal distributed, neither could the P_i's. But according to central limit theory[9], when the data number m is large enough, once the error terms do not differ from normal distribution too much, the distributions of b_i's are still normal.

The least square estimators for **b** are unbiased:

$$\mathbf{E}[\mathbf{b}] = \beta \tag{3.5}$$

The variance-covariance matrix $\sigma^2[\mathbf{b}]$ is given by:

$$\sigma^2[\mathbf{b}] = \begin{bmatrix} \sigma^2[b_0] & \sigma[b_0,b_1] & \cdots & \sigma^2[b_0,b_k] \\ \sigma[b_1,b_0] & \sigma^2[b_1] & \cdots & \sigma[b_1,b_k] \\ \vdots & \vdots & \ddots & \vdots \\ \sigma[b_k,b_0] & \sigma[b_k,b_1] & \cdots & \sigma^2[b_k] \end{bmatrix} = \sigma^2 \cdot (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1} \tag{3.6}$$

The estimated variance-covariance matrix $\mathbf{s}^2[\mathbf{b}]$ is given by:

$$\mathbf{s}^2[\mathbf{b}] = \begin{bmatrix} s^2[b_0] & s[b_0,b_1] & \cdots & s^2[b_0,b_k] \\ s[b_1,b_0] & s^2[b_1] & \cdots & s[b_1,b_k] \\ \vdots & \vdots & \ddots & \vdots \\ s[b_k,b_0] & s[b_k,b_1] & \cdots & s^2[b_k] \end{bmatrix} = MSE \cdot (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1} \tag{3.7}$$

For the normal error regression model, we have:

$$\frac{b_i - \beta_i}{s[b_i]} \sim t(m-k-1), \quad i = 0,1,\ldots,k \tag{3.8}$$

Where $t(m\text{-}k\text{-}1)$ is the t distribution with degree of freedom of $(m\text{-}k\text{-}1)$. Figure 3.4 shows the shape of the t distribution. In the follow, $t(1\text{-}\alpha/2;\ m\text{-}k\text{-}1)$ denotes the $(1\text{-}\alpha/2)\times 100$ percentile of the appropriate t distribution.

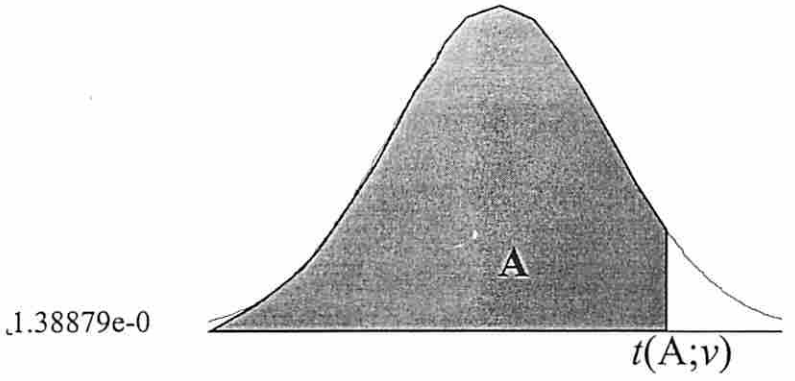


Figure 3.4 The t distribution

The confidence interval for single coefficient β_i with a confidence level of $(1\text{-}\alpha)$ is:

$$[\ b_i - t(1-\alpha/2;m-k-1)\cdot s[b_i],\ \ b_i + t(1-\alpha/2;m-k-1)\cdot s[b_i]\ ] \tag{3.9}$$

which means that, after the estimated value b_i is given, the probability that the "real" coefficient β_i is in the confidence interval is $1\text{-}\alpha$. Given a confidence level(e.g., 95%), small confidence intervals for coefficients implies that the macro-model is well designed and well trained.

At a certain confidence level $1\text{-}\alpha$, we can also make the decision that β_i is not zero if

$$\left|\frac{b_i}{s[b_i]}\right| > t(1-\alpha/2;n-k-1) \tag{3.10}$$

This is yet another test for variable reduction. If we have already used the variable reduction algorithm to select the variables, it is not necessary to do this test.

3.4.2 Inference about prediction of new observations

Information about the estimation error is the key factor for us 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 vector pair and give confidence interval of the prediction for a given confidence level as detailed next.

Values of regression variables, X_{new}, 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 $\hat{P}_{new}$ for the vector pair. At this point, we do not know the actual value of this 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 confidence interval of the observation P_{new} is given by:

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

where $s[P_{new}]$ is standard deviation of the new observation which is given by:

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

In simple terms, the probability that the absolute value of the difference between $\hat{P}_{new}$ and P_{new} exceeds t(1-α/2; m-k-1)·$s[P_{new}]$, is α.

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, we 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 equation(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. Table 3.3 gives the experimental results of error computation for C1908 general purpose macro-model at confidence level of 95%.

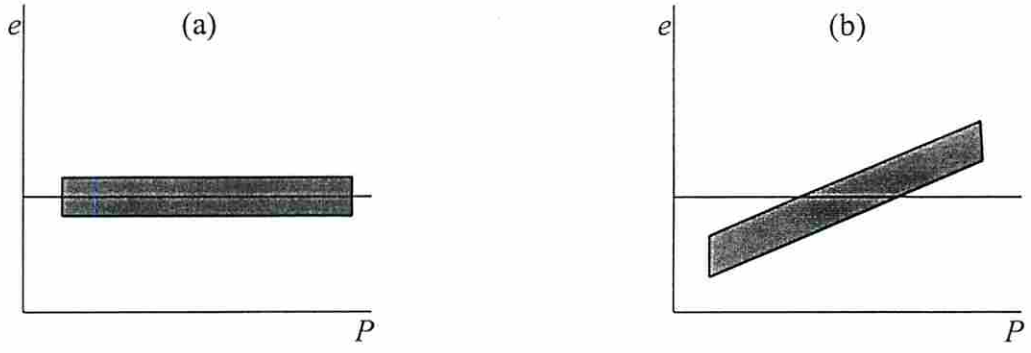
Table 3.3 Experimental results of error confidence interval prediction (mW)

Vec. Pair	Fitted Power	Confidence Interval	Actual Power	Correct Error Prediction?
1	1.302	[0.737, 1.868]	1.537	YES
2	1.323	[0.78, 1.867]	1.944	NO
3	1.329	[0.764, 1.895]	1.213	YES
4	1.274	[0.703, 1.845]	1.499	YES
5	1.765	[1.23, 2.299]	1.676	YES
6	3.095	[2.561, 3.63]	2.808	YES
7	1.994	[1.46, 2.528]	2.325	YES

Since the fitted power value follows normal distribution, the average error is approximately 1/4 of the confidence interval.

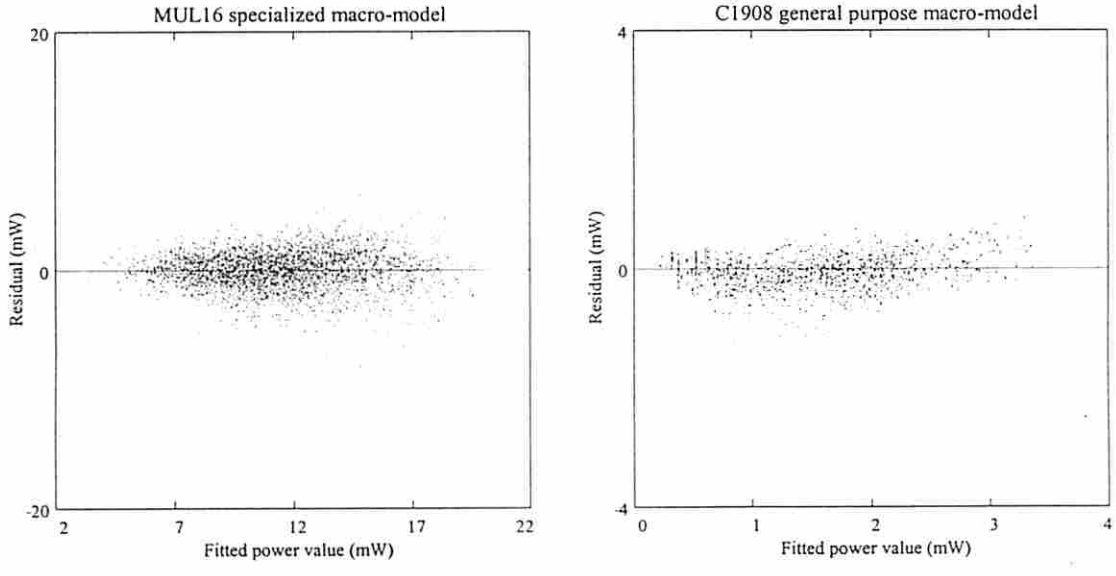
3.4.3 Model verification

After the model is fitted to all the training units, we must go back and check the foundation of the methodology. The correctness of the form of macro-model is the aspect we need to look into. Whether a linear macro-model is appropriate for the training sequence being analyzed can be studied in several ways. One of them is a *residual plot against the fitted power values*. The following are two typical prototype residual plots.



Prototype(a) suggests that the macro-model is appropriate for the training set, which means that macro-model power P is a linear statistical relation of the variables we have chosen. However prototype(b) doesn't suggest the linearity of the macro-model.

The residual plots for some of our macro-models are shown below.



The plots show that the macro-models we have built, both specialized and general purpose, exhibit a linear statistical relation.

Section 4. Experimental results and discussion

Some experimental results are shown in Table 4.1.

Table 4.1 Experimental** results for some specialized and general purpose macro-model***

Module	Type	# of var.'s	Average Error	Sum Error
MUL16	Specialized	44	10.6%	4.3%
C6288	Specialized	44	7.9%	3.1%
ADD16	Gen. Purp.*	64	8.0%	1.0%
MUL4	Gen. Purp.*	80	9.4%	1.2%
C1355	Gen. Purp.*	82	13.3%	10.9%
C1908	Gen. Purp.*	66	15.4%	2.3%
C3540	Gen. Purp.*	78	15.5%	5.2%

* "Gen. Purp." Stands for "General Purpose".

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

*** We are currently in the process of generating the specialized macro-models for more complicated cores such as Kalman Filter and Viterbi Decoder. The results will be presented at the final submission.

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

$$P = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k \tag{4.1}$$

Then the one-shot macro-model for average power estimation is:

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

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

$$\overline{P} = \beta_0 + \beta_1 \mathrm{Prob}(X_1) + \beta_2 \mathrm{Prob}(X_2) + \cdots + \beta_k \mathrm{Prob}(X_k) \tag{4.3}$$

There exist difficulties to build one-shot macro-model directly using the signal/transition probability as variables. Because a long input sequence is only one unit in the training set. That will cause problems such as lack of information, insufficient training, analysis difficulty, etc. Comparing to it, cycle-based macro-model has advantage in the aspects of larger design space of variable selection, adequate data for training set design, available methodology for error control, etc.

Section 5. Conclusions

In the paper, we introduced the cycle-based macro-model for RT-level power estimation. In this way we are able to estimate not only the average power consumption at RT-level, but also the power distribution over all the cycles that being estimated. The macro-model is built on the basis of regression analysis. Two variable selection strategies were discussed: specialized and general purpose. The number of variables can be reduced using statistical tests. The statistical methodology enable 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 certain accuracy.

References

[1] S.Powell and P. Chau, Estimating power dissipation of VLSI signal processing chips: The PFA techniques, Proceedings of IEEE Workshop on VLSI Signal Processing IV, volume IV, p.250-259, 1990.

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

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

[4] Jan M. Rabaey, P. Landman, Activity-sensitive architectural power analysis for the control path, Proceedings of International Symposium on Low Power Design, p.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, p.294-299, Nov. 1994.

[6] John Neter, W. Wasserman, M.H. Kutner, Applied Linear Regression 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, p.696-701, Jun. 1996.

[8] C. Ding, etal., Stratified Random Sampling for Power Estimation, to appear in ICCAD 96.

[9] Allen T. Craig Robert, V. Hogg, Introduction to Mathematical Statistics, Fourth Edition, Macmillan Publishing, 1978.