Cycle-Accurate Macro-Models for RT-Level Power Analysis

Qing Wu, *Member, IEEE*, Qinru Qiu, Massoud Pedram, *Member, IEEE*, and Chih-Shun Ding, *Member, IEEE*

***Abstract* — In this paper we present a methodology and techniques for generating cycle-accurate macro-models for RT-level power analysis. The proposed macro-model predicts not only the cycle-by-cycle power consumption of a module, but also the moving average of power consumption and the power profile of the module over time. We propose an exact power function and approximation steps to generate our power macro-model. First order temporal correlations and spatial correlations of up to order 3 are considered in order to improve the estimation accuracy. A variable reduction algorithm is designed to eliminate the "insignificant" variables using a statistical sensitivity test. Population stratification is employed to increase the model fidelity. Experimental results show our macro-models with 15 or fewer variables, exhibit <5% error for average power and <20% errors for cycle-by-cycle power estimation compared to circuit simulation results using Powermill.**

***Keywords* — cycle-accurate, macro-model, power estimation, VLSI, CMOS, low power, statistical, regression**

I. Introduction

Due to rapid progress in the semiconductor manufacturing, the device density and operating frequency have greatly increased, making power consumption a major design concern. High power consumption exacerbates the reliability problem by raising the die temperature and by increasing current density on the supply rails. It also reduces the battery life which is a key concern in portable devices. Therefore, low power design requirements are driving a new breed of computer aided design methodologies and tools which in turn rely on accurate and efficient estimation tools at various design abstraction levels.

Power estimation at RT level is crucial in achieving a short design cycle. The standard hierarchical simulation approach to RT-level power estimation consists of three steps: 1) functionally simulate the RT-level description and collect the input sequences for each circuit block. 2) simulate each block at gate or circuit-level using the collected input sequences. 3) add the power consumption for all blocks to produce the power consumption of the whole circuit. The disadvantage of this approach is that it requires the interaction between RT-level simulators and low-level simulators and that power evaluation is actually done at gate-level or circuit-level where the simulation speed is low.

Alternatively, one could use the macro-modeling technique for power estimation at RT-level. In this technique, low-level simulations of modules under their respective input sequence is replaced by power macro-model equation evaluation (which can be performed very fast).

Macro-modeling techniques use capacitance models for circuit modules and activity profiles for data or control signals [1-5]. 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. To improve the accuracy, more sophisticated macro-model equations have been proposed. Dual Bit Type model, proposed in [2], exploits the fact that, in the data path or memory modules, switching activities of high order bits depend on the temporal correlation of data while lower order bits behave similarly to white noise data. Thus a module is completely characterized by its capacitance models in the MSB and LSB regions. The break-point between the two regions is determined based on the signal statistics collected from simulation runs. The Activity-Based Control (ABC) model [4] is proposed to estimate the power consumption of random-logic controllers. An Input-Output model has been proposed in [5] to capture the relation between power and input signal probability, input transition density, and output transition density. By introducing variables related to output activity, the Input-Output model improves the estimation accuracy compared to the models which do not make use of the output information. One common feature of the above macro-model techniques is that, they only provide information about average power consumption over a relatively large number of clock cycles.

The above techniques, which are suitable for estimating the average-power dissipation, are referred to as *cumulative power macro-models*. In some applications, however, estimation of average power only is not sufficient. Other important tasks include the estimation of the k-cycle moving average of the power, power profiling on a cycle-by-cycle basis, and estimation of the rate of current change from one cycle to next. This information is crucial for circuit reliability (maximum current limits, heat dissipation and temperature gradient calculation, latch-up conditions) analysis, DC/AC noise analysis (DC drop and inductive bounce on power and ground lines), and design optimization (power/ground net topology, construction and sizing, number and placement of decoupling capacitors, buffer insertion, etc.). For example, the k-cycle average power can provide power consumption information for any given window of time. To perform these tasks requires knowing the power consumption value for every clock cycle. If the macro-modeling technique does not provide such information, the circuit designers will have to resort to gate-level or circuit-level simulation again. Consequently, cumulative macro-models are considered to have limited use.

The notions of a *pattern-dependent macro-model* [6][7] and a *cycle-accurate macro-model* [8] are related. In the following, we describe cycle-accurate macro-models as initially described in [8]. Let P_k denote the power consumption of some module in clock cycle k, then we can write:

$$P_k = F(V_{k-1}, V_k) \tag{1.1}$$

where V_k and V_{k-1} denote the input vectors applying to the module at cycles k and k-1, and F is some function of the input vector pairs. Note that the above definition for cycle-accurate macro-model is based on the assumptions that there is no glitching at the inputs of

the module and that all input transitions arrive at the same time. These assumptions are valid in synchronous designs. In addition, (1.1) assumes that the influence of the states of the floating nodes within gates can be ignored, that is, the power dissipation of a combinational module only depends on a pair of consecutive vectors. This assumption holds in most cases of interest.

The goal of power macro-modeling is to find function *F*, given an input vector sequence *V* (the so called *training set*) for the module and given the corresponding power consumption values.

In this paper, we propose the methodology of building cycle-accurate macro-models for circuit modules. Compared to previous work, our approach makes the following tangible contributions:

1. The macro-model generated by our approach can predict the cycle-based power consumption, as well as the average power. Our cycle-accurate macro-model equation can be easily transformed into a cumulative macro-model equation.
2. We present an exact power consumption function which captures the relation between the circuit power consumption and the spatial-temporal correlations of primary inputs. This is used as the starting point for our macro-model generation. It can be a useful guide for generating macro-models for other purpose.
3. We introduce piecewise linear power macro-model equations to increase the fidelity of the macro-model.
4. A statistical significance test is proposed to eliminate "insignificant variables" and therefore simplify the macro-model equation.
5. Because of the statistical nature of our macro-model, it can be validated and improved by statistical methods. The estimation error can be predicted for given confidence level.

Experimental results show that, our cycle-accurate macro-models having 15 or fewer variables, exhibit <5% error in average power, and <20% error in cycle power compared to circuit simulation results using Powermill [9].

This paper is organized as follows. Section II gives the theoretical background for regression analysis. Section III discusses a procedure of building the macro-model whereas Section IV presents the experimental results. Section V is the conclusion of our work.

II. BACKGROUND

A. Introduction to linear regression analysis

We define a cycle-accurate power macro-model as a linear function between estimated power dissipation of a vector pair and the characteristic values of the vector pair, that is, we write:

$$P = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k \qquad (2.1)$$

where $\hat{P}$ is the predicted power dissipation, $\beta_0, \beta_1, \cdots, \beta_k$ 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. The methods for extracting values of $X_1, X_2, \cdots, X_k$ will be discussed in Section 3.1. The regression parameters are calculated by doing least-squares-fit during the linear regression analysis.

Based on the theory of linear regression analysis [10], we can define the relation between the actual power *P* (e.g. the power value simulated by Powermill) and the estimated power as:

$$P = \hat{P} + \varepsilon = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon \qquad (2.2)$$

where ε is called the residual term of the linear regression model and follows a normal distribution with mean value 0 and variance σ^2. Equation (2.2) means that *P* is a random variable which follows a normal distribution with mean value $\hat{P}$ and variance σ^2.

Assume that we have been given the equation form of the macro-model as in Eqn.(2.1) and have performed Powermill simulations (observations) on *m* randomly sampled vector pairs in the population (this set of *m* vector pairs is referred to as the *training set*) so that we have obtained *m* simulation results (observation values) of power consumption. The linear regression model for vector pairs from the training set can be written as:

$$P_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_k x_{i,k} + \varepsilon_i, \quad i = 1,2,\cdots,m \qquad (2.3)$$

or in matrix form as:

$$\mathbf{P} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \qquad (2.4)$$

where P_i's are random variables corresponding to observations: $(x_{i,1}, x_{i,2}, \cdots, x_{i,k})$ $i = 1,2,..,m$; $\beta_0, \beta_1, \cdots, \beta_k$ are the regression coefficients; $x_{i,1}, x_{i,2}, \cdots, x_{i,k}$ are known values derived from the input vector pair $(V_{i,1}, V_{i,2})$; and ε_i's are independent random variates representing deviation from the mean value of power with variance $\mathbf{VAR}[\varepsilon_i] = \sigma^2$, and $\mathbf{Cov}[\varepsilon_i, \varepsilon_j] = 0$, for $i \neq j$. Consequently, the random vector $\mathbf{P}$ has an expected value of $\mathbf{E}[\mathbf{P}] = \mathbf{X}\boldsymbol{\beta}$ and the variance-covariance matrix of $\mathbf{P}$ is $\mathbf{Cov}[\mathbf{P}] = \sigma^2\mathbf{I}$ where $\mathbf{I}$ is the identity matrix.

The $\boldsymbol{\beta}$ coefficients are estimated using the least squares estimator by substituting the actual power values for $\mathbf{P}$:

$$\mathbf{b} = (\mathbf{X}^{\mathrm{T}} \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^{\mathrm{T}} \cdot \mathbf{P} \qquad (2.5)$$

where

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

It has been proven in [10] that the least squares estimator is an unbiased estimator for $\boldsymbol{\beta}$, i.e., $\mathrm{E}[\mathbf{b}] = \boldsymbol{\beta}$. The estimated (fitted) power from macro-model is given by the multiplication of input variables and the estimated coefficients:

$$\hat{\mathbf{P}} = [\hat{P}_1, \hat{P}_2, \cdots, \hat{P}_m] = \mathbf{X}\mathbf{b} \qquad (2.7)$$

In the following, we define some relevant terms for regression analysis:

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

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

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

regression mean squares: $MSR = SSR/k$

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

B. Evaluating the quality of a macro-model

The quality of the macro-models can be evaluated in terms of the following criteria:

1. Correlation factor: From the coefficient of multiple correlation *R*, we derive a similar quantity *r* as:

$$r = 1/(1-R^2) = 1 + SSR/SSE \qquad (2.8)$$

We call *r* the correlation factor of the macro-model. *r* is a monotonic-increasing function of *R*. In many applications of linear regression, *r* is a general measure of the quality of a regression model since it represents linearity of the model and the magnitude of the error. It also reflects the stability or fidelity of a macro-model. The higher the *r* value, the better the quality of the regression model. The *r* value may differ from one population to next for the same macro-model. Therefore, the *r* values of different

macro-models should be compared only when they are subjected to the same input population.

2. Errors: Error in cycle power (ECP) gives the average error when estimating power on cycle by cycle basis while error in average power (EAP) gives the average error when estimating the average power. More precisely, we can write:

$$ECP = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{P}_i - P_i}{P_i}\right|, \qquad EAP = \frac{\sum_{i=1}^{n}\hat{P}_i - \sum_{i=1}^{n}P_i}{\sum_{i=1}^{n}P_i} \qquad (2.9)$$

In most cases the training set only represents a small portion of the target population, and in some extreme cases, the training set will be totally different from the population. Designed based on such training sets, the macro-models can exhibit good quality on the training set and on populations which have similar characteristics as the training set, but not on other populations. To assess the quality of macro-model, accuracy comparison of macro-models should be carried out on populations (the set of input vector pairs and corresponding Powermill power values) whose behavior is different from that of the training set. On the other hand, the design of training set is very important. The more closely the training set represents the target population, the more accurate the resulting macro-model will be. In addition, it is very difficult to predict in advance what type of population that macro-model will be subjected to. Therefore, careful design of a good macro-model equation form is the key to reducing the estimation error. Last, but not least, extracting the $X_1, X_2, \cdots, X_k$ variables in Eqn.(2.1) from the input vector pair is important in designing a good macro-model.

III. Macro-model Construction

The overall macro-model generation procedure is described in Figure 1. The macro-model generation procedure consists of four major steps: variable selection, training set design, variable reduction, and least squares fit. Notice that the macro-model equation design refers to not only the form of the equation (linear function, values of coefficients), but also the procedure for extracting variable values from the input vector pair.

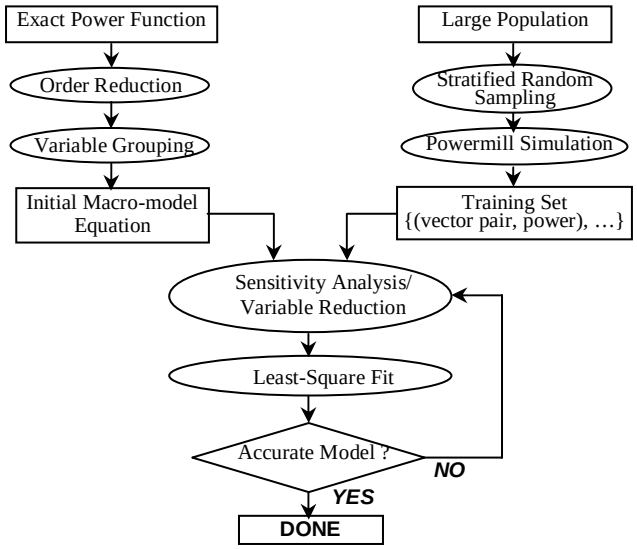


Figure 1 The workflow of generating a cycle-accurate macro-model

In the variable selection step, we start with an exact function that relates the cycle power and the input vector pair. The function terms are organized according to the order of spatial correlations between bits of the input vector pair. During order reduction, high order terms are dropped based on a performance-cost trade-off consideration. Variable grouping is performed to collapse the variables which have similar influence on power into the same groups. At the end of this step, we obtain an initial macro-model equation which is a reduced form of the original exact function and define the procedure for extracting variable values from the input vector pair.

The training set design step is simple and straight forward compared to other parts. It starts from a very large set of vector pairs, which is in turn obtained either from real application or is synthetically generated. Stratified random sampling is performed to obtain a much smaller sub-set of vector pairs which is representative of the original set. Finally, the module under analysis is simulated using Powermill and by applying the vector pairs in the sub-set. The vector pairs in the sub-set and their power values form the so called training set.

In the third step of the flow, a statistical variable reduction algorithm is applied on the initial macro-model equation using the training set. The goal of this algorithm is to eliminate the variables which have the least impact on the circuit power dissipation, and therefore, limit the number of variables in the final macro-model equation. Subsequently, we obtain a final macro-model equation consisting of the most power-significant variables, that is, we obtain the final macro-model equation in the form of Eqn.(2.1) with a relatively small number of variables ($k \leq 15$).

In the fourth step of the flow, the training set is used once again to form the linear regression model in Eqn.(2.2). By using Eqn.(2.5), least-squares fit is performed to calculate the regression parameters of the macro-model. Power estimation for the training set is done using Eqn.(2.7). Model evaluation should be carried out before the macro-model is used in real applications. The standards for evaluating the quality of a macro-model were discussed in the previous section.

In our approach, to improve the fidelity of macro-model, we build different macro-models for different ranges of variable values. We call this procedure the population stratification approach (cf. Section 3.2) and the resulting macro-model the piecewise linear macro-model.

In the remainder of this section, we will focus on discussing variable selection and variable reduction steps which are the key procedures for building a good macro-model. Other steps either will be discussed briefly, or have been addressed in previous paragraphs.

A. Variable selection

1) The exact functional relation between the cycle power and the input vector pair

If we ignore the influence of states of the floating nodes within gates (which is relatively small) on the circuit power, we can write:

$$P = f(\vec{t}_1, \vec{t}_2, \cdots, \vec{t}_k) \qquad (3.1)$$

where k is the number of primary inputs (notice that this "k" is different from the "k" that was used in Section II) and $\vec{t}_1, \vec{t}_2, \cdots, \vec{t}_k$ are the so called transition variables which are encoded by a bit vector as follows:

$$\begin{aligned} &\vec{t}_i = [a \quad b \quad c], && i = 1,2,\cdots,k \\ &a=0, b=0, c=0 && \text{if input } i: 0 \to 0 \\ &a=1, b=0, c=0 && \text{if input } i: 0 \to 1 \\ &a=0, b=1, c=0 && \text{if input } i: 1 \to 0 \\ &a=0, b=0, c=1 && \text{if input } i: 1 \to 1 \end{aligned} \qquad (3.2)$$

Note that only 3 of these transitions are independent. Also we use a 3-bit encoding scheme instead of a 2-bit encoding (which is the

minimum length encoding) because the 3-bit encoding is more suitable for expressing the exact power function.

Define the ⊗ operation and + operation (normal addition) between two vectors as follows:

$$[u_1,u_2,\cdots,u_m]\otimes[v_1,v_2,\cdots,v_l]=[u_1v_1,u_1v_2,\cdots,u_1v_l,u_2v_1,u_2v_2,\cdots,u_2v_l,\cdots,u_mv_1,u_mv_2,\cdots,u_mv_l] \quad (3.3)$$

$$[u_1,u_2,\cdots,u_m]+[v_1,v_2,\cdots,v_m]=[u_1+v_1,u_2+v_2,\cdots,u_m+v_m]$$

We give the exact functional relation between the cycle power and the input vector pair (the transition variables) as:

$$P=a_0+\sum_{i=1}^{k}\vec{t}_i\cdot\begin{bmatrix}a_i^{0\to1}\\a_i^{1\to0}\\a_i^{1\to1}\end{bmatrix}+\sum_{i=1}^{k}\sum_{j=i+1}^{k}\vec{t}_i\otimes\vec{t}_j\cdot\begin{bmatrix}a_{i,j}^{0\to1,0\to1}\\a_{i,j}^{0\to1,1\to0}\\\vdots\\a_{i,j}^{1\to1,1\to1}\end{bmatrix}$$

$$+\cdots+\vec{t}_1\otimes\vec{t}_2\otimes\cdots\otimes\vec{t}_k\cdot\begin{bmatrix}a_{1,2,\ldots,k}^{0\to1,0\to1,\cdots,0\to1,0\to1}\\a_{1,2,\ldots,k}^{0\to1,0\to1,\cdots,0\to1,1\to0}\\\vdots\\a_{1,2,\ldots,k}^{1\to1,1\to1,\cdots,1\to1,1\to1}\end{bmatrix} \quad (3.4)$$

$$=a_0+\sum_{i=1}^{k}\vec{t}_i\cdot\vec{a}_i^{\mathrm{T}}+\sum_{i=1}^{k}\sum_{j=i+1}^{k}\vec{t}_i\otimes\vec{t}_j\cdot\vec{a}_{i,j}^{\mathrm{T}}+\cdots+\vec{t}_1\otimes\vec{t}_2\otimes\cdots\otimes\vec{t}_n\cdot\vec{a}_{1,2,\ldots,k}^{\mathrm{T}}$$

where $\vec{t}_i$ is called *order* 1 *transition variable* of input *i*, $\vec{t}_i\otimes\vec{t}_j$ is called *order* 2 *joint transition variable* of inputs *i* and *j*, etc., and $\vec{a}_i^{\mathrm{T}},\cdots,\vec{a}_{1,2,\ldots,k}^{\mathrm{T}}$ are vectors of real numbers representing the variable coefficients. Entries of the vector variables are either 0 or 1 and the sum of entries in each vector adds up to 1. Notice that each vector variable includes multiple scalar variables, when we are talking about the "number of variables" in the function, we refer to the number of scalar variables. Notice that an order *i* joint transition variable describes the absence (value = 0) or presence (value = 1) of the corresponding bit-level transitions at the inputs of the module.

Example: Given an input vector pair (101→011) with bit 1 to bit 3 listed from left to right, $\vec{t}_1=\{010\}$, $\vec{t}_2=\{100\}$, $\vec{t}_3=\{001\}$, $\vec{t}_1\otimes\vec{t}_2=\{000100000\}$, $\vec{t}_1\otimes\vec{t}_3=\{000001000\}$, $\vec{t}_2\otimes\vec{t}_3=\{001000000\}$, $\vec{t}_1\otimes\vec{t}_2\otimes\vec{t}_3=\{000000000001000000000000000\}$.

The power consumption calculated by Eqn.(3.4) is:

$$P=a_0+a_1^{1\to0}+a_2^{0\to1}+a_3^{1\to1}+a_{1,2}^{1\to0,0\to1}+a_{1,3}^{1\to0,1\to1}+a_{2,3}^{0\to1,1\to1}+a_{1,2,3}^{1\to0,0\to1,1\to1}.$$

Theorem 1 *Equation (3.4) gives the exact power consumption for any vector pair applied to the inputs of any combinational module with k inputs. Furthermore, coefficients in the equation are unique for a given module.* [1]

Note that Theorem 1 does not mean that two different modules cannot have the same coefficients.

2) Transitive fanout correlation between primary inputs

It is obvious that a_0 = 0 since power consumption for vector pair (00…0)→(00…0) must be zero. All other coefficients in Eqn.(3.4) can be uniquely determined from circuit-level simulation on some specific vector pairs.

For example, to compute coefficient $a_1^{0\to1}$, we simulate the module using the vector pair {(0,0…0), (1,0…0)} and obtain the power consumption value of $P_1^{0\to1}$. From equation (3.4) we know:

$$P_1^{0\to1}=f(t_1^{0\to1},t_2^{0\to0},\cdots,t_k^{0\to0})=a_1^{0\to1}$$

To compute $a_{1,2}^{0\to1,0\to1}$, we simulate the module using the vector pair {(0,0…0), (1,1,0…0)} and obtain the power consumption value of $P_{1,2}^{0\to1,0\to1}$. Again from Eqn.(3.4) we know:

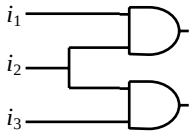
$$P_{1,2}^{0\to1,0\to1}=f(t_1^{0\to1},t_2^{0\to1},t_3^{0\to0},\cdots,t_k^{0\to0})=a_1^{0\to1}+a_2^{0\to1}+a_{1,2}^{0\to1,0\to1}$$

$$\Rightarrow a_{1,2}^{0\to1,0\to1}=P_{1,2}^{0\to1,0\to1}-P_1^{0\to1}-P_2^{0\to1}$$

and so on.

Definition Inputs $i_1,i_2,\ldots,i_j$ are *transitive fanout correlated* when their transitive fanout cones in the circuit have at least one common node, that is, there exists at least one node (internal node or output) of the module whose logic function includes all inputs $i_1,i_2,\ldots,i_j$. *j* is called the *order* of the correlation.

For the sake of simplicity, we use "correlation" to stand for "transitive fanout correlation" in the remainder of this paper.



(a)

order 1 corr.	order 2 corr.	order 3 corr.
i_1; i_2; i_3	(i_1, i_2); (i_2, i_3)	None

(b)

Figure 2 Example of Transitive Fanout Correlation

Figure 2(a) shows a simple 3-input 2-gate circuit. Since it has only 3 inputs, the highest possible order of correlation between inputs is 3. The table in (b) shows the correlated inputs for different orders. Notice that the input pair (i_1, i_3) is not on the list of order 2 correlations and triplet (i_1,i_2,i_3) is not in the list of order 3 correlations because the corresponding inputs have no common nodes among their transitive fanout cones.

The coefficients in Eqn.(3.4) essentially reflect the correlation between the corresponding (joint) transition probabilities and the power consumption in a circuit.

Proposition 1 *If $i_1,i_2,\ldots,i_j$ are not correlated, all entries of $\vec{a}_{i_1,i_2,\cdots,i_j}$ are zero.*

Corollary *If J is the highest order of correlation among inputs of a module, the first J+1 terms of Eqn.(3.4) are sufficient to model the exact power for any input vector pair applied to the module.*

3) Function order reduction

Eqn.(3.4) gives an exact representation of the relation between power and input transition. However, this form is too complicated for practical use. In this section, we will discuss the first step to simplify the macro-model function, which is order reduction.

From Eqn.(3.4), we know that the complexity of the macro-model increases exponentially with the order of the input correlation we which consider. Evidently, ignoring the high order term leads to some estimation error. The first question, is how much error will be introduced if we drop certain high order terms. The second question is what the cost will be if we keep more terms in the original function. Table 1 shows some examples of the percentage error caused by ignoring the high order input correlations. Column 1 gives the circuit name. Circuit A is a 4-bit multiplier, B is a 4-bit ripple carry adder without carry in, and C is an 8-input random logic circuit. In the experiment, all the coefficients of the exact power function in Eqn.(3.4) are calculated in the way discussed in section 3.1.2. Then we do power calculation on the population of all possible (4^k, *k* is the number of inputs) input vector pairs to the module using the reduced-order functions of Eqn(3.4), i.e., ignoring the high order terms (by assuming that the coefficients of high order transition variables to be all zero). The power values calculated by the reduced-order functions are compared with the actual power values. The average relative errors are then reported

[1] All proofs are omitted for the sake of brevity. Please refer to [11] for detailed proofs.

in Table 1 from the 1st data column to the 8th data column. The integer number *i* on top of each column indicates the maximum order to which the function terms are kept. For example, in the 4th data column, number "4" means that, the reduced-order function is the same as Eqn.(3.4) except that transition variables (and their coefficients) with order higher that 4 are ignored. The last row of the table shows the total number of variables in the reduced-order functions for each of the circuits.

Table 1 Average percentage error in power dissipation when using reduced-order functions

Circuit	1	2	3	4	5	6	7	8
A	99.8%	42.6%	31.7%	23.9%	13.7%	4.1%	0.8%	0.0%
B	40.3%	9.2%	11.9%	9.6%	8.4%	4.3%	1.3%	0.0%
C	37.6%	13.1%	8.9%	7.7%	5.1%	4.3%	1.0%	0.0%
	24	276	1788	7458	21066	41478	58974	65535

Table 1 shows that keeping higher order terms tends to, although not monotonically, to provide more accurate power estimation results. One significant improvement shown in the table is from the 1st data column (keeping only order 1 terms) to the 3rd data column (keeping order 1, 2 and 3 terms). From this point on, the complexity of the reduced-order function increases much faster than the percentage error decreases. This observation is also supported by the experimental results which is presented in Section IV. We therefore approximate Eqn.(3.4) by ignoring terms with order higher than 3. The reduced-order function is written as:

$$P = a_0 + \sum_{i=1}^{k} \bar{t}_i \cdot \begin{bmatrix} a_i^{0\to1} \\ a_i^{1\to0} \\ a_i^{1\to1} \end{bmatrix} + \sum_{i=1}^{k}\sum_{j=i+1}^{k} \bar{t}_i \otimes \bar{t}_j \cdot \begin{bmatrix} a_{i,j}^{0\to1,0\to1} \\ a_{i,j}^{0\to1,1\to0} \\ \vdots \\ a_{i,j}^{1\to1,1\to1} \end{bmatrix}$$
$$+ \sum_{i=1}^{k}\sum_{j=i+1}^{k}\sum_{l=j+1}^{k} \bar{t}_i \otimes \bar{t}_j \otimes \bar{t}_l \cdot \begin{bmatrix} a_{i,j,l}^{0\to1,0\to1,0\to1} \\ a_{i,j,l}^{0\to1,0\to1,1\to0} \\ \vdots \\ a_{i,j,l}^{1\to1,1\to1,1\to1} \end{bmatrix} + \varepsilon \qquad (3.5)$$
$$= a_0 + \sum_{i=1}^{k} \bar{t}_i \cdot \bar{a}_i^{\mathrm{T}} + \sum_{i=1}^{k}\sum_{j=i+1}^{k} \bar{t}_i \otimes \bar{t}_j \cdot \bar{a}_{i,j}^{\mathrm{T}}$$
$$+ \sum_{i=1}^{k}\sum_{j=i+1}^{k}\sum_{l=j+1}^{k} \bar{t}_i \otimes \bar{t}_j \otimes \bar{t}_l \cdot \bar{a}_{i,j,l}^{\mathrm{T}} + \varepsilon$$

where ε is the error caused by approximation.

We can minimize error ε by re-computing the coefficient values by doing least-squares fit for Eqn.(3.5). However Eqn.(3.5) is too complicated to be our macro-model equation since the number of variables in it is $3\cdot k + 9\cdot C_k^2 + 27\cdot C_k^3$, which is too high! Furthermore, the use of 0-1 variables in (3.5) makes it very difficult to significantly reduce the number of variables using a regression approach.

4) Variable grouping

To further reduce the function complexity in Eqn.(3.5), we use a variable grouping approach as will be described next. This approach offers two advantages: 1) uses integer variables which are easier to work with and offer more flexibility compared to 0-1 variables, 2) has a constant number of variables which is independent of number of primary inputs, *k*.

We define G_1 as the set of all inputs, G_2 as the set of all possible combinations of two inputs, G_3 as the set of all possible combinations of three inputs:

$$G_1 = \{1,2,\cdots,k\},$$
$$G_2 = \{(1,2),(1,3),\cdots,(1,k),(2,3),\cdots,(k-1,k)\},$$
$$G_3 = \{(1,2,3),(1,2,4),\cdots,(1,2,k),(1,3,4),\cdots,(k-2,k-1,k)\}$$

Note that G_i consists of indices for order *i* transition variables. The variable grouping technique forms N_1 subsets of G_1, N_2 subsets of G_2, and N_3 subsets of G_3 such that:

$$\bigcap_{g=1}^{N_1} G_{1,g} = \phi, \quad \bigcup_{g=1}^{N_1} G_{1,g} \subseteq G_1, \quad \bigcap_{g=1}^{N_2} G_{2,g} = \phi, \quad \bigcup_{g=1}^{N_2} G_{2,g} \subseteq G_2,$$
$$|G_{1,g}| \le K_1 \qquad |G_{2,g}| \le K_2$$
$$\bigcap_{g=1}^{N_3} G_{3,g} = \phi, \quad \bigcup_{g=1}^{N_3} G_{3,g} \subseteq G_3 \qquad (3.6)$$
$$|G_{3,g}| \le K_3$$

where K_1, K_2, K_3 bounds are given. The size constraints are specified to manage the complexity of macro-model equation characterization and evaluation.

We approximate equation (3.5) by assuming that:

$$\begin{bmatrix} a_i^{0\to1} \\ a_i^{1\to0} \\ a_i^{1\to1} \end{bmatrix}_{3\times1} \equiv \begin{bmatrix} b_{1,g}^{0\to1} \\ b_{1,g}^{1\to0} \\ b_{1,g}^{1\to1} \end{bmatrix}_{3\times1} \qquad \forall i \in G_{1,g}, \quad g = 1,2,\cdots,N_1$$

$$\begin{bmatrix} a_{i,j}^{0\to1,0\to1} \\ a_{i,j}^{0\to1,1\to0} \\ \vdots \\ a_{i,j}^{1\to1,1\to1} \end{bmatrix}_{9\times1} \equiv \begin{bmatrix} b_{2,g}^{0\to1,0\to1} \\ b_{2,g}^{0\to1,1\to0} \\ \vdots \\ b_{2,g}^{0\to1,1\to1} \end{bmatrix}_{9\times1} \qquad \forall (i,j) \in G_{2,g}, \ g = 1,2,\cdots,N_2$$

$$\begin{bmatrix} a_{i,j,l}^{0\to1,0\to1,0\to1} \\ a_{i,j,l}^{0\to1,0\to1,1\to0} \\ \vdots \\ a_{i,j,l}^{1\to1,1\to1,1\to1} \end{bmatrix}_{27\times1} \equiv \begin{bmatrix} b_{3,g}^{0\to1,0\to1,0\to1} \\ b_{3,g}^{0\to1,0\to1,1\to0} \\ \vdots \\ b_{3,g}^{0\to1,1\to1,1\to1} \end{bmatrix}_{27\times1} \qquad \forall (i,j,l) \in G_{3,g}, g = 1,2,\cdots,N_3$$

where $b_{\{1,2,3\},g}^{\cdots}$ are constant real numbers. To minimize the error introduced by the above approximation, we should do a careful variable grouping. We first calculate the coefficients for terms of orders 1,2 and 3 in Eqn.(3.4) by using the method discussed in Section 3.1.2. Next, we calculate a set of order 1, 2 and 3 c-values as shown below:

$$c_i = \frac{1}{3}(a_i^{0\to1} + a_i^{1\to0} + a_i^{1\to1}), \quad i = 1,2,\cdots,k$$

$$c_{i,j} = \frac{1}{9}(a_{i,j}^{0\to1,0\to1} + a_{i,j}^{0\to1,1\to0} + \cdots + a_{i,j}^{1\to1,1\to1}), \quad i,j = 1,2,\cdots,k, \ i < j$$

$$c_{i,j,l} = \frac{1}{27}(a_{i,j,l}^{0\to1,0\to1,0\to1} + a_{i,j,l}^{0\to1,0\to1,1\to0} + \cdots + a_{i,j,l}^{1\to1,1\to1,1\to1}), \quad i,j,l = 1,2,\cdots,k, \ i < j < l$$

Transition variables of order *i* are sorted in increasing order of the corresponding *c*-values for order *i*. The domain of the *c*-values is divided into several sub-domains such that the number of transition variables with *c*-values in different sub-domains is approximately equal, but is less than the corresponding K_i values. The indices of transition variables with *c*-values in different sub-domains define the groups. The first N_1 groups with the largest absolute *c*-values are used to form $G_{1,g}$ (g=1,2,…,N_1) as defined in Eqn.(3.6). Other groups with smaller *c*-values are abandoned. Similarly, the first N_2 or N_3 groups with the largest absolute *c*-values are adopted as $G_{2,g}$ or $G_{3,g}$ (g=1,2,…,N_2 or N_3). Notice that this is a heuristic grouping algorithm and other techniques may be used as well.

Example Let N_1 = 3, N_2 = 4, N_3 = 2. Assume we want to do variable grouping for a macro-model equation corresponding to a 6-input circuit. The group size constraints are set as: K_1= K_2= K_3 = 3. Firstly we calculate c_i (i = 1,2,…,6) values. Assume, they are given by:

$$c_1 = 0,\ c_2 = 0.5,\ c_3 = 1.9,\ c_4 = 2.0,\ c_5 = 0,\ c_6 = 0$$

We divide the domain of c-values, [0, 2.0], into 3 sub-domains: [0, 0.5), [0.5, 1.5), and [1.5, 2.0].

The grouping for single inputs is: $G_{1,1} = \{1,5,6\}, G_{1,2} = \{2\}, G_{1,3} = \{3,4\}$. We keep $G_{1,1}$ through $G_{1,3}$.

Then we compute the value of $c_{i,j}$ (i,j = 1,2,…,k, i < j). Again, suppose:

$$c_{1,2} = 1.0,\ c_{1,3} = 0.3,\ c_{1,4} = -0.2,\ c_{1,5} = -0.4,\ c_{1,6} = 1.6,\ c_{2,3} = 1.6,$$
$$c_{2,4} = 0.8,\ c_{2,5} = 1.4, c_{2,6} = 0.7,\ c_{3,4} = -1.1,\ c_{3,5} = 0.2,\ c_{3,6} = 0.2,$$
$$c_{4,5} = 1.6,\ c_{4,6} = -0.7,\ c_{5,6} = 0.9$$

The division of c-values is: [-1.1, -0.5), [-0.5, 0), [0, 0.5), [0.5, 1.0), [1.0, 1.5), and [1.5, 1.6]

The grouping for input pairs is:

$$G_{2,1} = \{(1,6),(2,3),(4,5)\},\ G_{2,2} = \{(2,5),(1,2)\}$$
$$G_{2,3} = \{(2,4),(2,6),(5,6)\},\ G_{2,4} = \{(1,3),(3,5),(3,6)\},$$
$$G_{2,5} = \{(1,4),(1,5)\},\ G_{2,6} = \{(3,4),(4,6)\}$$

We only keep $G_{2,1}$, $G_{2,2}$, $G_{2,3}$, and $G_{2,6}$,

The case for grouping variables of order 3 is solved similarly.

Let's introduce some notation:

$T_{1,g}^{i\to j}$: the total number of transitions of type $i\to j$ in group $G_{1,g}$

$T_{2,g}^{i\to j,k\to l}$: the total number of pair-wise joint transitions of type ($i\to j$ $k\to l$) in group $G_{2,g}$

$T_{3,g}^{i\to j,k\to l,m\to n}$: the total number of joint transitions of type ($i\to j$, $k\to l$, $m\to n$) in group $G_{3,g}$

$$\vec{T}_{1,g} = \sum_{i\in G_{1,g}} \vec{t}_i = \left[T_{1,g}^{0\to1} \quad T_{1,g}^{1\to0} \quad T_{1,g}^{1\to1}\right]_{1\times3}$$

$$\vec{T}_{2,g} = \sum_{(i,j)\in G_{2,g}} \vec{t}_i \otimes \vec{t}_j = \left[T_{2,g}^{0\to1,0\to1} \quad T_{2,g}^{0\to1,1\to0} \quad \cdots \quad T_{2,g}^{1\to1,1\to1}\right]_{1\times9}$$

$$\vec{T}_{3,g} = \sum_{(i,j,l)\in G_{3,g}} \vec{t}_i \otimes \vec{t}_j \otimes \vec{t}_l = \left[T_{3,g}^{0\to1,0\to1,0\to1} \quad T_{3,g}^{0\to1,0\to1,1\to0} \quad \cdots \quad T_{3,g}^{1\to1,1\to1,1\to1}\right]_{1\times27}$$

We can thus write our initial cycle-accurate macro-model as in equation (3.7).

$$\begin{aligned} P = b_0 &+ \sum_{g=1}^{N_1}\left[T_{1,g}^{0\to1} \quad T_{1,g}^{1\to0} \quad T_{1,g}^{1\to1}\right]\cdot\begin{bmatrix} b_{1,g}^{0\to1} \\ b_{1,g}^{1\to0} \\ b_{1,g}^{1\to1}\end{bmatrix} \\ &+ \sum_{g=1}^{N_2}\left[T_{2,g}^{0\to1,0\to1} \quad T_{2,g}^{0\to1,1\to0} \quad \cdots \quad T_{2,g}^{1\to1,1\to1}\right]\cdot\begin{bmatrix} b_{2,g}^{0\to1,0\to1} \\ b_{2,g}^{0\to0,1\to0} \\ \vdots \\ b_{2,g}^{1\to1,1\to1}\end{bmatrix} \\ &+ \sum_{g=1}^{N_3}\left[T_{3,g}^{0\to1,0\to1,0\to1} \quad T_{3,g}^{0\to1,0\to1,1\to0} \quad \cdots \quad T_{3,g}^{1\to1,1\to1,1\to1}\right]\cdot\begin{bmatrix} b_{3,g}^{0\to1,0\to1,0\to1} \\ b_{3,g}^{0\to0,0\to1,1\to0} \\ \vdots \\ b_{3,g}^{1\to1,1\to1,1\to1}\end{bmatrix} \end{aligned} \tag{3.7}$$

In terms of N_1, N_2, N_3, values, the number of variables in the macro-model is $3N_1$+$9N_2$+$27N_3$, which is independent of the number of circuit inputs k.

Table 2 shows the experimental results for three macro-models using different number of groups and using different grouping strategies. For Macro-model 1, N_1 = 1, N_2 = 1, N_3 = 1; For Macro-model 2, N_1 = 8, N_2 = 8, N_3 = 2, and the single inputs, input pairs, and input triplets are grouped randomly; For Macro-model 3, N_1 = 8, N_2 = 8, N_3 = 2, and our variable grouping heuristic is used. The input sequence is randomly generated.

Table 2 Experimental results for variable grouping

Module	Macro-model 1		Macro-model 2		Macro-model 3	
	R	ECP (%)	r	ECP (%)	r	ECP (%)
C1355	1.98	8.07	1.57	9.19	2.54	7.76
C1908	1.41	15.36	1.42	15.04	2.76	11.16
C2670	1.18	11.66	1.19	11.64	1.63	10.38
C3540	1.42	17.47	1.76	15.48	2.37	12.19
C432	1.11	29.07	1.12	29.00	2.46	20.15
C5315	1.21	9.87	1.30	9.4	2.79	8.1
C6288	2.15	8.10	2.47	7.6	2.79	6.82
C7552	1.04	33.00	1.11	30.94	6.39	9.24
C880	1.42	19.82	1.31	20.78	1.95	16.34
Mul16	2.34	8.90	2.57	8.32	2.96	7.04
Adder16	2.05	8.63	2.12	8.44	4.08	6.15

Results show that macro-models 1 and 2 have similar correlation factors and ECP errors, while the quality of macro-model 3 is clearly better than the other two.

From Table 2, we can draw the following conclusions:

- Using more groups in variable grouping improves the quality of macro-models.
- A good variable grouping technique is very important to obtain a high quality macro-model.

B. Population stratification

From our experiments we found that the regression factor r between the estimated power and the actual power is different for different ranges of power dissipation. This means that the regression model is not strictly linear over the range of all possible power values. This phenomenon occurs in many practical situations. One reason for the lack of linearity is that the macro-model equation is only an approximation to the power-transition function. During the variable selection, we discard the high order terms in the power-transition function and group subsets of variables of given order together. The approximation introduces some non-linearity into the macro-model equation. This effect is more pronounced when the number of variables is small.

To improve the quality of our macro-model, we refine the macro-model to a *piece-wise linear regression model*. At the first step, we stratify the training set into several disjoint subsets (strata) based on the switching activity of the vector pairs in the training set. A vector pair will fall into exact one of these strata. Then the macro-model is trained separately for each subset of the training set. When we apply this piece-wise linear macro-model to estimate the power for a given vector pair, we first examine the switching activity range of the vector pair, and then invoke the macro-model equation which was trained using vector pairs with a similar switching activity.

Theorem 2 The correlation measure r_{str} of the macro-model obtained by the population stratification is no worse than that without population stratification r_{nostr}, i.e.,

$$r_{str} \geq r_{nostr}$$

Experimental results in Table 3 shows the improvement on the regression factor r of the macro-model with the population stratification approach (Macro-model 1) and without it (Macro-

model 2). We use Eqn.(3.7) as the macro-model equation, and the input sequence contains both biased (non-random, wider range of switching activity and power) and random vectors.

Table 3 Experimental results for the stratification approach

Module	Macro-model (w/ str.)		Macro-model (w/o str.)	
	r	ECP (%)	*r*	ECP (%)
C1355	26.0	7.86	16.4	8.76
C1908	12.8	9.34	10.4	11.19
C2670	23.6	8.77	20.0	10.22
C3540	19.7	11.45	11.8	12.88
C432	6.5	19.07	5.3	22.96
C5315	27.9	7.64	26.8	8.72
C6288	46.5	6.03	37.0	7.16
C7552	43.7	6.58	39.0	7.36
C880	10.4	14.19	10.4	15.32
Mul16	30.0	6.32	27.9	6.90
ADDER16	38.1	5.64	18.4	6.73

Notice that population stratification can be done not only according to the switching activity of the input vector pair, but also according to the transition behavior of some special inputs such as clock, mode control, etc. However, this is not the subject of this paper.

C. Variable reduction

In the initial macro-model equation (3.7), the number of variables is about 150. Although the large number of variables improves the quality of the macro-model, we would like to avoid evaluating a large macro-model equation for every clock cycle. Therefore, we must reduce the number of variables in the equation without incurring a large error.

In our approach, the modified forward stepwise regression procedure [10] is used to reduce the number of variables. The search method develops a sequence of regression models. At each step, one *X* variable is added to or deleted from the final macro-model equation. The criterion used for adding or deleting variables is the F^* *statistics* of the regression theory. The algorithm is described next:

Input of the algorithm: Given are a set of candidate variables { X_1, X_2, …, X_n } which is in the initial macro-model, a training set (values of variables for input vector pair and corresponding Powermill power value), a low threshold t_0 for deleting a variable, a high threshold t_1 for adding a variable, an upper bound of number of variables MAX_{var}, *S* is the set of selected variables.

Step 0 (Initialization) : Set $S = \phi$ and $C = \{ X_1, X_2, \ldots, X_n \}$

Step 1 (Find the first variable) : Fit a one-variable linear regression model for each variable X_i in *C*. The F^* test for each model is given by:

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

where *MSR* and *MSE* were defined in Section II. 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.

This step finds the first variable for the final macro-model. The F^* test is used to find the "most significant" variable (as far as power dissipation in the module is concerned).

Step 2 (Add a variable) : Assume $S = \{\}$, 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:

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

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.

This step adds one more variable into the final macro-model. The F^* test is used to find the "most significant" variable to add to the set of existing (already selected) variables.

Step 3 (delete a 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*:

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

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

After adding a new variable into the macro-model, the "significance" of some old variable may be reduced due to the joint effect of the newly added variable and other old variables. In such a case, we have to remove the old variable from the macro-model. The F^* test is used to find the "most insignificant" variable to delete from the set of existing (already selected) variables.

Step 4 : Repeat Steps 2 and 3 until one of following conditions is true:

1. Algorithm terminates in Step 2.
2. $C=\phi$.
3. The number of variables in *S* equals to MAX_{var}.

In our approach, the number of variables in the candidate set is 162 at the beginning (since we set N_1=8, N_2=8, and N_3=2). We choose $t_0 = t_1 = 10.0$, $MAX_{var} = 15$. For most macro-models, the algorithm terminated at the 3rd condition at step 4 when the number of variables equals to MAX_{var}. Only for one of the macro-models the algorithm terminated at step 2 when $F_j^* < t_1$.

IV. EXPERIMENTAL RESULTS

We have built our cycle-accurate macro-models for several modules, including the ISCAS-89 benchmarks. In our macro-models, we also included variables representing transitions on circuit outputs, but only for two of the circuits (C432 and C880) variables related to outputs survived the variable reduction phase.

The experimental setup is as follows. For each circuit, the population size is set to 80,000 vector pairs (including both random and non-random sub-sequences). We first simulate each circuit for the entire sequence using Powermill and record the cycle-by-cycle power. Size of the training set is set to 3,000 . The macro-model is then trained using the training set. After the macro-model is built, we apply it to different subsets of the population. These subsets are selected such that their power behaviors are different from that of the training set. Average ECP and EAP are computed by averaging the ECP's and EAP's of all sub-sets. The correlation factor *r* is computed based on the fitted results on the entire population. Experimental results for our cycle-accurate macro-models is summarized in Table 4.

Experimental results shows that our macro-model technique are very accurate when estimating power consumption at RT-level. The average ECP and EAP are 10.2% and 2.0%, respectively. Meanwhile, if we compare the results with those of macro–model 1

in Table 3, which is the full-length macro-model before variable reduction, we see that our variable reduction algorithm is significantly reducing the number of variables without incurring large error.

Table 4 Experimental results for cycle-accurate macro-models

Circuit	No. of Variables	*r*	ECP (%)	EAP (%)
C1355	15	13.2	9.3	2.7
C1908	15	7.9	11.6	2.0
C2670	15	19.8	9.6	2.0
C3540	15	9.7	12.5	2.0
C432	14	5.1	19.3	3.1
C5315	15	27.4	7.8	1.6
C6288	15	45.9	6.2	1.9
C7552	15	6.58	6.9	1.1
C880	15	8.7	14.3	3.2
Mul16	15	34.3	6.5	1.6
ADDER16	15	28.8	6.4	1.1
Average Error			10.2	2.0

V. Conclusion

In conclusion of our work, we present a method for generating cycle-accurate macro-models for RT-level power analysis. The proposed macro-model predicts not only the cycle-by-cycle power consumption of a module, but also the moving average of power consumption and the power profile of the module over time. We present an exact power consumption function to derive our final macro-model equation. A variable reduction algorithm has been proposed to eliminate the "insignificant" variables based on statistical sensitivity test. First order temporal correlations and spatial correlations of up to order 3 are considered in order to improve the estimation accuracy. Population stratification has been used to increase the fidelity of the macro-model. Experimental results show that, the macro-models have 15 or fewer variables and exhibit <5% error in average power, and <20% errors in cycle-by-cycle power compared to circuit simulation results using Powermill.

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, "Architectural Power Analysis: The Dual Bit Type Method", *IEEE trans. VLSI*, Vol. 3, pp. 173-187, Jun. 1995.

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

[4] P. Landman and J. Rabaey, "Activity-Sensitive Architectural Power Analysis", *IEEE trans. CAD*, Vol. 15, pp. 571-587, Jun. 1996.

[5] S. Gupta and F.F. Najm, "Power Macromodeling for High Level Power Estimation", *Proceedings of Design Automation Conference*, pp.365-370, Jun. 1997

[6] H.Mehta, R.M.Owens and M.J.Irwin, "Energy Characterization Based on Clustering", *Proceedings of ACM/IEEE Design Automation Conference*, pp. 702-707, Jun. 1996.

[7] L.Benini, A.Bogliolo, M.Favalli and G.De Micheli, "Regression models for behavioral power estimation", *International Workshop on Power, Timing, Modeling, Optimization and Simulation*, pp. 179-188, Sep.1996.

[8] Q. Wu, C. Ding, C. Hsieh, and M. Pedram, "Statistical Design of Macro-models For RT-Level Power Evaluation", *Proceedings of the Asia and South Pacific Design Automation Conference*, pp.523-528, Jan. 1997.

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

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

[11] Q. Wu, Q. Qiu, M. Pedram and C. Ding, "Cycle-Accurate Macro-Modeling: Algorithm and Implementation", University of Southern California, CENG Technical Report No. 98-11, May 1998.