Energy and Peak-Current Per-Cycle Estimation at RTL†

Subodh Gupta
Cadence Design Systems Inc.
555 River Oaks Parkway
San Jose, California 95134, USA

and

Farid N. Najm
ECE Department
University of Toronto
Toronto, Ontario M5S-3G4, Canada

Abstract – We present novel macromodeling techniques for estimating the energy dissipated and peak-current drawn in a logic circuit for every input vector pair (we call this the *energy-per-cycle* and *peak-current-per-cycle* respectively). The macromodels are based on classifying the input vector pairs on the basis of their Hamming distances and using a different equation-based macromodel for every Hamming distance. The variables of our macromodel are the zero-delay transition counts at three logic *levels* inside the circuit. We present an automatic characterization process by which such macromodels can be constructed. The energy-per-cycle macromodel provides a transient energy waveform, and can also be used to estimate the moving average energy over any time window, whereas peak-current-per-cycle macromodel provides peak-current which can be used for studying IR drop problems. Some key features of this technique are: 1) the models are compact (linear in the number of inputs), 2) they can be used for *any* input sequence, and 3) the characterization is automatic and requires no user intervention. These approaches have been implemented and models have been built and tested for many circuits. The average errors observed in estimating the energy-per-cycle and peak-current-per-cycle are under 20%. The energy-per-cycle model can also be used to measure the long-term average power, with an observed error of under 10% on average.

*This research was supported in part by the National Science Foundation (NSF MIP 97-10235) and by the Semiconductor Research Corporation (SRC 97-DJ-484 and 99-TJ-682), with technical mentorship from Texas Instruments, IBM, and Intel.

1. Introduction

With the advent of portable and high-density micro-electronic devices, the power dissipation of very large scale integrated (VLSI) circuits has become a critical concern. Modern microprocessors are hot, and their power consumption can exceed 30 or 50 Watts. Due to limited battery life, reliability issues, and packaging/cooling costs, power consumption has become a more critical design concern than speed and area in some applications. Hence to avoid problems associated with excessive power consumption, there is a need for CAD tools to help in estimating the power consumption of VLSI designs.

A number of CAD techniques have been proposed for gate-level power estimation (see [1] for a survey). However, by the time the design has been specified down to the gate level, it may be too late or too expensive to go back and fix high power problems. Hence in order to avoid costly redesign steps, power estimation tools are required that can estimate the power consumption at a high level of abstraction, such as when the circuit is represented only by the Boolean equations. This will provide the designer with more flexibility to explore design trade-offs early in the design process, reducing the design cost and time.

In response to this need, a number of high-level power estimation techniques have been recently proposed (see [2] for a survey). Two styles of techniques have been proposed, which we refer to as top-down and bottom-up. In the top-down techniques [3, 4], a combinational circuit is specified only as a Boolean function, with no information on the circuit structure, number of gates/nodes, etc. Top-down methods are useful when one is designing a logic block that was not previously designed, so that its internal structural details are unknown.

In contrast, bottom-up methods [5–10, 12, 17–20] are useful when one is reusing a previously designed logic block, so that all the internal structural details of the circuit are known. In this case, one develops a *power macromodel* for this block which can be used during high-level power estimation (of the overall system in which this block is used), in order to estimate the power dissipation of this block without performing a more expensive gate-level power estimation on it.

The methods in [5, 6, 8, 10, 12, 16, 17, 18, 19] target the average power over a long time period. However, in many applications, the average power may not be enough. Indeed, it is often important to know the instantaneous power dissipation, as a time-waveform, i.e., what one may refer to as a *transient power* waveform. The most important application where the transient power waveform is needed is probably the analysis of the power and ground bus networks for finding IR-drop problems which lead to reduced circuit speed due to the reduced power supply voltage. Another obvious application is noise analysis, because glitches on the power supply are coupled into the circuit leading to noisy and possibly erroneous signals. This is especially important in circuits that are designed with power-down or sleep modes. When parts of these circuits are turned on or off, large supply current transients will result and it is important to understand the noise and IR-drop implications of these transients. How can one produce a high-level macromodel for a logic block that reports the power supply current waveform for every possible vector stimulus? This is very difficult because the number of vectors and the variability in the shapes of all the different waveforms is very large. A simpler problem (assuming a clocked system) would to be to provide some important data points of this current waveform, in order to have some understanding about the current waveform. We have considered the average and peak (or maximum) to be the two significant characteristics of the current waveform, which can be used for studying moving average energy and IR-drop problem, respectively, over any time window. This requires building macromodels that give the energy dissipated and peak-current drawn in the circuit due to a given vector pair, effectively the energy-per-cycle and peak-current-per-cycle respectively. For estimating energy-per-cycle at RTL some solutions [7, 9] have been proposed, but none has been proposed for estimating peak-current-per-cycle.

The method in [7] characterizes the power dissipation of circuits based on input transitions rather than input statistics. Since the number of possible input transitions for an n-input combinational circuit is 2^{2n}, they present a clustering algorithm to compress the input transitions into clusters of input transitions that have the same power values (approximately). They use heuristics to implement the clustering algorithm, but it is not clear how efficient the method would be on large circuits.

In [9], the authors presented a macromodel for estimating the cycle-by-cycle power at the RTL. The proposed methodology consists of three steps: module equation form generation and variable selection, variable reduction, and population stratifications. The generated macromodel has 15 variables. They show good accuracy in estimating average and cycle-by-cycle power. The macromodels are dependent on a training vector set, so that the accuracy is compromised if the training set is not similar to the vector set to be applied.

In this paper, we present a novel macromodeling approach that provides the energy-per-cycle and peak-current-per-cycle without running the risk of a combinatorial explosion and which has good accuracy even when the applied vectors are different from the characterization vectors set. We classify the input vector pairs on the basis of their Hamming distances and use a different equation-based macromodel for every Hamming distance. The variables of our macromodel are the zero-delay transition counts at three logic *levels* (see section 2.1 for definition). The energy-per-cycle equation-based macromodel consist of 10 and 5 coefficients for real-delay and zero-delay energy respectively. Moreover, this energy-per-cycle macromodel can be used for estimating the average energy of the circuit over any specified time period. In the case of peak-current-per-cycle we make only real-delay macromodel, which consists of 10 coefficients. This paper is an extended version of [24].

This paper is organized as follows. In next section, we describe the approach for estimating energy-per-cycle. In section 3, we extend this approach for estimating peak-current-per-cycle. In section 4, we give the characterization flow of our method. In section 5, we present results and some applications of our macromodels and finally in section 6, we give some conclusions.

2. ENERGY-PER-CYCLE

We assume that a circuit block is given that is described at a low level of abstraction (say, at the gate level). We assume this circuit is clocked and, for simplicity, we assume that a single clock drives all the memory elements (registers or flip-flops), but this is not a limitation of this technique. Upon every new cycle of the clock a new primary input vector is applied, and the combinational part of this circuit block is presented with a new logic vector $\mathbf{x}_i$. In general, $\mathbf{x}_i$ consists of primary input bits and of state bits. We are required to build a macromodel for this circuit that gives the energy consumed in every clock cycle, given the primary inputs vector sequence.

We assume that the macromodel is intended to be used in a high-level analysis in which the transient power dissipation characteristics of this block are to be examined under some vector stimulus. As part of this analysis, this block will be simulated to determine its outputs. By a "high-level" of abstraction, we mean that the simulation is at higher than a gate level. Specifically, for purposes of this high level analysis, we assume that the the circuit block under consideration would be specified as sets of Boolean functions (representing combinational logic blocks) that exist between banks of clocked memory elements (flip-flops). This level of abstraction is sometimes referred to as a *structural RTL*, i.e. a Register Transfer Level description that includes complete information on the memory elements but only functional (Boolean) information about everything else.

Since the inputs and outputs of the flip-flops are known, it is easy to estimate the energy-per-cycle due to each flip-flop, using some cell-level model for them. The main difficulty lies in modeling the energy-per-cycle for the combinational logic parts so that their energy and can be found without having to perform detailed gate-level simulation. For this reason, in the remainder of this paper, we will focus on combinational circuits.

Consider a combinational circuit with N nodes. Let C_i be the capacitance associated with node i and $n_i\left(\mathbf{x}_1, \mathbf{x}_2\right)$ be the number of transitions at node i due to the input vector pair $(\mathbf{x}_1, \mathbf{x}_2)$ (we use **bold** letters to denote vector quantities). Then, the energy dissipated for the input vector pair $(\mathbf{x}_1, \mathbf{x}_2)$ is given by:

$$E\left(\mathbf{x}_1, \mathbf{x}_2\right) = 0.5V_{dd}^2 \sum_{i=1}^{N} C_i n_i \left(\mathbf{x}_1, \mathbf{x}_2\right) \qquad (1)$$

We refer to $E(\mathbf{x}_1, \mathbf{x}_2)$ as the energy-per-cycle. A brute-force way of modeling $E(\mathbf{x}_1, \mathbf{x}_2)$ is to simulate the circuit, say at the gate-level, for all possible input vectors and store the energy value corresponding to each vector pair $(\mathbf{x}_1, \mathbf{x}_2)$ in a look-up table. But for a circuit with M primary inputs, the total number of possible input vector pairs is 4^M which grows exponentially with M. Hence, the complexity of this approach is exponential in M ($\mathcal{O}(4^M)$), making it practically infeasible for all but the smallest circuits.

Therefore, the goal of macromodeling is to find a function $\hat{E}\left(\mathbf{x}_1, \mathbf{x}_2\right)$ which would be a good approximation to (1) over all possible input vector pairs

$(\mathbf{x}_1, \mathbf{x}_2)$ and which would be less complex. Our approach, which aims to achieve this, is a two step process:

Step 1. Identify a number of variables $v_1(\mathbf{x}_1, \mathbf{x}_2), v_2(\mathbf{x}_1, \mathbf{x}_2), \cdots, v_L(\mathbf{x}_1, \mathbf{x}_2)$ which best represent the dependence of the energy-per-cycle on the vector pair $(\mathbf{x}_1, \mathbf{x}_2)$. We call this step *variable selection* and is described in section 2.1.

Step 2. If M is the number of bits in the vectors $\mathbf{x}_i$, let $h \in \{0, 1, \cdots, M\}$, be the Hamming distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ (i.e., h is the number of bits that are different). Then, choose a polynomial model (linear or quadratic) $\hat{E}_h(\mathbf{x}_1, \mathbf{x}_2)$, for every Hamming distance, in terms of the variables chosen in step 1. Determine the coefficients of the model using the method of Recursive Least Squares (RLS) [18,21]. We call this step *macromodel construction* and is described in section 2.2.

Therefore, our final macromodel is:

$$\hat{E}_h(\mathbf{x}_1, \mathbf{x}_2) = f_h(v_1(\mathbf{x}_1, \mathbf{x}_2), v_2(\mathbf{x}_1, \mathbf{x}_2), \cdots, v_L(\mathbf{x}_1, \mathbf{x}_2)) \tag{2}$$

Since the Hamming distance can take M possible values (0 is not considered, as the energy-per-cycle is zero for no input transition), we will have M such macromodels. We will show later in section 4, that the complexity of our approach is linear in M $(\mathcal{O}(M))$.

For example, if $L = 1$ and $f_h(\cdot)$ is a linear function, then the macromodel becomes:

$$E_h(\mathbf{x}_1, \mathbf{x}_2) = c_0(h) + c_1(h)\, v_1(\mathbf{x}_1, \mathbf{x}_2) \tag{3}$$

where $c_0(h)$ and $c_1(h)$ are the coefficients for the corresponding Hamming distance h, found using RLS [18,21].

In the next section we will describe an approach for choosing the variables $v_i(\mathbf{x}_1, \mathbf{x}_2)$.

2.1 *Variable Selection*

A combinational circuit can always be *levelized* so that its gates are tagged with the *level* values that represent their distance from the primary inputs. Thus every gate whose inputs are all primary inputs is said to have level 1. Every other gates whose inputs are either outputs of level 1 gates or are primary inputs is said to have level 2, etc. The levelization algorithm [20] has linear time complexity and is standard in most logic/timing simulation systems. The largest level number K used in levelizing a circuit is called the circuit *depth*. Moreover, the output nodes of a gate have the same level number as that of the gate. By grouping the nodes which are at the same level, (1) can be rewritten as:

$$E(\mathbf{x}_1, \mathbf{x}_2) = 0.5 V_{dd}^2 \sum_{i=1}^{K} \sum_{j=1}^{G_i} C_j n_j(\mathbf{x}_1, \mathbf{x}_2) \tag{4}$$

where G_i is number of nodes that are outputs of gates at level i. Moreover, the G_is satisfy the following condition:

$$N = \sum_{i=1}^{K} G_i \tag{5}$$

Since we are trying to estimate the energy-per-cycle at RTL, some approximations seem inevitable. We start with the simplifying assumption that the capacitance of a node at a certain level is approximately equal to the average capacitance of all the nodes at that level. Therefore, (4) modifies to:

$$E(\mathbf{x}_1, \mathbf{x}_2) \approx 0.5 V_{dd}^2 \sum_{i=1}^{K} \mathcal{Q}_i \sum_{j=1}^{G_i} n_j(\mathbf{x}_1, \mathbf{x}_2) \tag{6}$$

where

$$\mathcal{Q}_i = \frac{1}{G_i} \sum_{j=1}^{G_i} C_j \tag{7}$$

It turns out that this is a very good approximation in practice, as seen in Fig. 1 which show a scatter plot of energy-per-cycle obtained from (6) (x-axis) and that obtained from (4) (y-axis), for c6288, one of the ISCAS-85 [13] benchmark circuits. The number of input vector pairs in the plot are 15000, which were generated randomly and the energy-per-cycle was estimated using [14]. It can be seen from the figure that energy-per-cycle values correlate very well and this behavior was observed for all the ISCAS-85 [13] benchmark circuits, which supports the approximation made in (6).

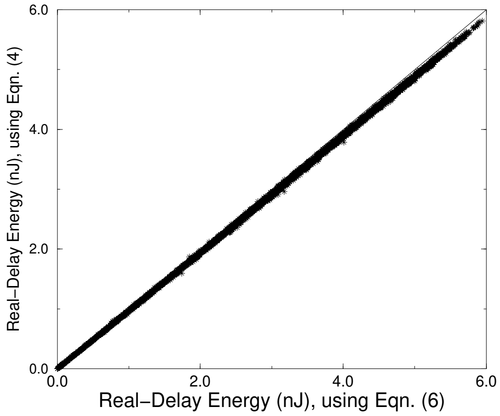


Figure 1. Plot of energy-per-cycle from (6) and (4), for c6288.

Substituting $\mathcal{N}_i(\mathbf{x}_1, \mathbf{x}_2)$ for $\sum_{j=1}^{G_i} n_j(\mathbf{x}_1, \mathbf{x}_2)$, (6) can be rewritten as:

$$E(\mathbf{x}_1, \mathbf{x}_2) \approx 0.5V_{dd}^2 \sum_{i=1}^{K} \mathcal{Q}_i \mathcal{N}_i(\mathbf{x}_1, \mathbf{x}_2) \qquad (8)$$

In (8), $\mathcal{Q}_i$ is known, as it can be obtained from the gate-level net-list and stored in a look-up table, to be used at RTL. But $\mathcal{N}_i(\mathbf{x}_1, \mathbf{x}_2)$ in (8) is unknown at RTL, as determining it requires the real-delay simulation of the whole circuit, which is prohibitive at RTL. One possibility is to estimate the real-delay energy-per-cycle from the zero-delay transitions at every level, i.e., from a simulation of the circuit that uses a zero-delay model for all the gates. In this case the macromodel would be:

$$\hat{E}(\mathbf{x}_1, \mathbf{x}_2) \approx 0.5V_{dd}^2 \sum_{i=1}^{K} \mathcal{Q}_i \mathcal{N}_i^z(\mathbf{x}_1, \mathbf{x}_2) \qquad (9)$$

where the superscript z signifies that the transitions are measured from a zero-delay simulation. To check the accuracy of this macromodel, input vector pairs were randomly generated and energy-per-cycle, $E(\mathbf{x}_1, \mathbf{x}_2)$, was estimated using [14] which also provides an estimate of $N_i^z(\mathbf{x}_1, \mathbf{x}_2)$. Using this, $\hat{E}(\mathbf{x}_1, \mathbf{x}_2)$, was also estimated using (9) and the relative error between the two energy values was computed. Table 1 shows the average of this error, for the ISCAS-85 [13] benchmark circuits, which is computed as:

$$\text{Avg. Error} = \frac{1}{P} \sum_{i=1}^{P} \frac{|E_i(\mathbf{x}_1, \mathbf{x}_2) - \hat{E}_i(\mathbf{x}_1, \mathbf{x}_2)|}{E_i(\mathbf{x}_1, \mathbf{x}_2)} \qquad (10)$$

where P is the number of input vector pairs (in this case, we used $P = 100,000$).

Table 1. Error in estimating real-delay energy-per-cycle using macromodel given by (9)

Circuit	*Avg.Error*	Circuit	*Avg.Error*
c499	76.49%	c5315	83.48%
c880	104.29%	c2670	104.28%
c1355	81.52%	c3540	103.49%
c1908	43.20%	c7552	86.22%
c432	92.13%	c6288	818.98%

It is clear from the table that the simple model of (9) is not good enough for estimating the real-delay energy-per-cycle as the glitches are not accounted for in (9). Another possibility is to construct the model as follows:

$$\hat{E}(\mathbf{x}_1, \mathbf{x}_2) = c_0 + \sum_{i=1}^{K} c_i \mathcal{N}_i^z(\mathbf{x}_1, \mathbf{x}_2) \qquad (11)$$

where the regression coefficients c_i would be determined using least squares fitting. Note that $\mathcal{Q}_i$ does not appear in (11), as it is contained in the regression coefficient c_i. In fact, we have found that the accuracy of (11) can be significantly improved if we generate different coefficients for every Hamming distance. One reason for this is that the energy per cycle depends strongly on the Hamming distance, as shown in Fig. 2 for c3540, an ISCAS-85 [13] benchmark circuit. Fig. 2 shows the energy-per-cycle for each Hamming distance, averaged over 1000 randomly generated input vector pairs for each Hamming distance.

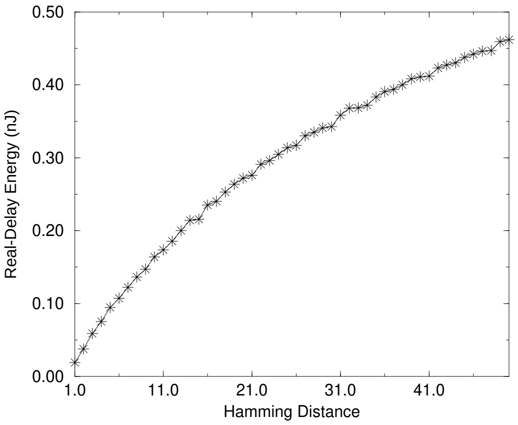


Figure 2. Showing the variation in energy for different Hamming distances, for c3540.

With this modification, the model of (11) becomes:

$$\hat{E}_h(\mathbf{x}_1, \mathbf{x}_2) = c_0(h) + \sum_{i=1}^{K} c_i(h) \mathcal{N}_i^z(\mathbf{x}_1, \mathbf{x}_2) \quad (12)$$

where the regression coefficients $c_i(h)$ are determined for each Hamming distance using RLS [21].

We call the macromodel of (12), the *golden model*. To check the accuracy of this model, $100,000$ input vector pairs were generated randomly. For each input vector pair, the actual energy-per-cycle was obtained using [14], which also provides estimate of $\mathcal{N}_i^z(\mathbf{x}_1, \mathbf{x}_2)$. Energy-per-cycle was also estimated using (12) and the relative error was computed for every input pair. Table 2, shows the average of this error, for ISCAS-85 [13] benchmark circuits, which is computed using (10) with $P = 100,000$. Also, shown in Fig. 3 is the real-delay energy-per-cycle waveform for c1908, an ISCAS-85 [13] benchmark circuit, where one trace was measured from simulation and the other was predicted from our model. The simulation was performed using a real-delay (not zero-delay) gate-level timing model, so that multiple transitions per cycle (glitches) were not ignored during the simulation. In order to generate this figure, we applied a low activity vector sequence for a while and then immediately applied a high activity vector sequence. While the agreement demonstrated in Fig. 3 is not exact (one would not expect that in a high-level model), it is clear that the accuracy is good enough to permit one to closely track the changes in power dissipation over time. Furthermore, the model has no time lag, it immediately reflects the change in power, which is a useful feature in practice. We consider this capability to be a major strength of this approach.

Table 2. Error in the *golden model* while estimating real-delay energy-per-cycle

Circuit	$Avg.Error$	Circuit	$Avg.Error$
c499	3.27%	c5315	8.25%
c880	9.18%	c2670	13.17%
c1355	5.76%	c3540	14.71%
c1908	8.91%	c7552	7.14%
c432	8.15%	c6288	13.33

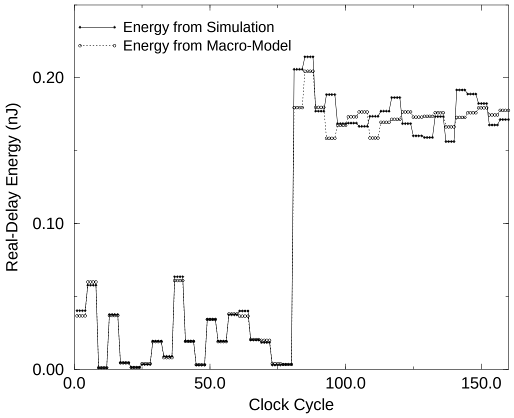


Figure 3. Energy waveform predicted using *golden model* for c1908.

Using the golden model requires one to perform a functional (zero-delay) simulation of the circuit, while monitoring the Boolean values at its internal nodes. Granted, for RTL simulation, we would have to simulate the circuit functionally anyway, but we normally would not evaluate the Boolean functions at every internal nodes. Thus the golden model probably requires more work than one is willing to do at RTL.

To resolve this problem, we propose to simplify (12) so that it does not require one to evaluate the Boolean functions at *all* the circuit nodes, but only at *some.* Specifically, we identify *three* logic levels inside the circuit, and require the user to measure the number of transitions only at the nodes in these levels. We have found experimentally, that by choosing only *three* levels, the percentage average error (10), over a large number of input vector pairs, was within 20% (see section 5.1), so that we lose only 5% in accuracy relative to our golden macromodel (12), for most of the circuits that we tested. Hence in our approach, $E(\mathbf{x}_1, \mathbf{x}_2)$ for the given input vector pair $(\mathbf{x}_1, \mathbf{x}_2)$, is determined from the zero-delay transitions at the chosen three levels. We use a *stepwise regression* procedure [23] to find these three levels. Stepwise regression is a well known variable selection method based on F^* *statistics* from regression theory [23].

Before explaining the algorithm, we begin with some useful terms for regression analysis:

1. Sum of squares error: $SSE = \sum_{j=1}^{P} \left(E_j(\mathbf{x}_1, \mathbf{x}_2) - \hat{E}_j(\mathbf{x}_1, \mathbf{x}_2) \right)^2$
2. $SSE(v_p, \ldots, v_q)$ is defined as SSE when $\hat{E}(\mathbf{x}_1, \mathbf{x}_2)$ is formulated as a regression equation on only the variables $v_p, \ldots, v_q$, where $v_i = \mathcal{N}_i^z(\mathbf{x}_1, \mathbf{x}_2)$ are the variables of the regression equation (12).
3. Mean squares error: $MSE(v_1, \ldots, v_k) = \frac{SSE(v_1, \ldots, v_k)}{P-k-1}$
4. Regression sum of squares: $SSR = \sum_{j=1}^{P} \left(\hat{E}_j(\mathbf{x}_1, \mathbf{x}_2) - \bar{E}(\mathbf{x}_1, \mathbf{x}_2) \right)^2$ where $\bar{E}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{P} \sum_{j=1}^{P} E_j(\mathbf{x}_1, \mathbf{x}_2)$.
5. $SSR(v_k \mid v_p, \cdots, v_q) = SSE(v_p, \cdots, v_q)$ $-SSE(v_p, \cdots, v_q, v_k)$.

Given the model (12), the aim of stepwise regression is to select 3 of the K variables $v_i = \mathcal{N}_i^z(\mathbf{x}_1, \mathbf{x}_2)$ that would be sufficient to compute $\hat{E}(\mathbf{x}_1, \mathbf{x}_2)$ with good accuracy. Stepwise regression is a heuristic procedure that considers only a limited number of the large $(2^k - 1)$ number of possibilities. It does this by iteratively adding (and removing) selected variables to (and from) a pool of candidate variables. The method is not optimal and is not flawless but is considered to be one of the best available. It is based on hypothesis testing, and requires one to select a *level of significance* which is used to check if a certain variable should be added to or removed from the pool. In [23], this is selected according to a percentile of the F-distribution, which depends on a specified level of confidence (we chose 95%) and on the number of variables in the pool. Since we are interested in selecting a pool of 3 variables only, the three resulting percentiles of the F-distribution are $F_1 = 3.84$, $F_2 = 3.00$, and $F_3 = 2.60$. Finally, we used $P = 500$ as the number of data points to be used for computing the regression coefficients and for computing SSE and the other statistics - this proved to be a good number to use in practice.

The flow of the stepwise regression procedure is as follows:

Step 1. Consider the possibility of using only a *single* variable $v_i = \mathcal{N}_i^z(\mathbf{x}_1, \mathbf{x}_2)$ in the regression equation. Find the regression coefficients and compute the errors in every case $v_1, v_2, \ldots, v_K$. For every case, compute the F^* statistic:

$$F_k^* = \frac{SSR(v_k)}{MSE(v_k)} \qquad (13)$$

The variable v_i with the largest F^* value is a candidate for addition to the pool. If this F^* exceeds a *threshold value* (F_1, in this case), the variable is added. Otherwise, terminate with failure.

Step 2. Assume v_i is the variable selected in step 1. Now calculate all regressions with *two* variables, with v_i being one of the pair, and compute F^* for each case:

$$F_k^* = \frac{SSR(v_k \mid v_i)}{MSE(v_i, v_k)} \qquad (14)$$

Choose the variable with largest F^* value as the candidate for addition at the second stage. If this F^* value exceeds a threshold value (F_2, in this case), the new variable is added. Otherwise, terminate with failure.

Step 3. Suppose v_j is added at the second stage. Now the stepwise regression routine examines whether any of the other variables already in the pool should be removed. For our illustration, there is at this stage only one other variable in the model, v_i, so that only one F^* statistic is obtained:

$$F_i^* = \frac{SSR\left(v_i \mid v_j\right)}{MSE\left(v_j, v_i\right)} \tag{15}$$

At later stages there would be a number of these F^* statistics, for each of the variables in the pool besides the last added, given all the other variables in the pool. The variable for which this F^* value is the smallest is the candidate for deletion. If this F^* value falls below a threshold value (either F_1, F_2, or F_3), the new variable is removed from the pool; otherwise it is retained.

Step 4. Suppose v_i is retained, so that both v_i and v_j are now in the pool. The stepwise regression routine now examines which variable is the next candidate for addition (repeat step 2), then examines whether any of the variables already in the pool should now be removed (repeat step 3), and so on until no further variables can be added or removed, at which point the search terminates. We actually terminate the search as soon as three variables have been added to the pool.

Let us denote the three variables chosen by the stepwise regression procedure, by $\mathcal{N}_{s1}^z(\mathbf{x}_1, \mathbf{x}_2)$, $\mathcal{N}_{s2}^z(\mathbf{x}_1, \mathbf{x}_2)$, and $\mathcal{N}_{s3}^z(\mathbf{x}_1, \mathbf{x}_2)$. In order to determine these variables at RTL, for the given input vector pair $(\mathbf{x}_1, \mathbf{x}_2)$, we have to perform fast functional simulation of the Boolean functions at the selected three levels. Note that we have employed a linear regression equation (12) in the stepwise regression procedure, in order to select the desired variables. We excluded cross-product terms and powers of the independent variables in order to keep the selection problem computationally inexpensive. However, one should keep in mind that he selection accuracy may be improved if one considers these additional terms.

Given any Hamming distance h, (2) now reduces to:

$$\hat{E}_h\left(\mathbf{x}_1, \mathbf{x}_2\right) = f_h\left(\mathcal{N}_{s1}^z\left(\mathbf{x}_1, \mathbf{x}_2\right), \mathcal{N}_{s2}^z\left(\mathbf{x}_1, \mathbf{x}_2\right) \mathcal{N}_{s3}^z\left(\mathbf{x}_1, \mathbf{x}_2\right)\right) \tag{16}$$

where the function $f_h(\cdot)$ is still unknown. In the next section, a methodology for determining this function will be presented.

2.2 *Macro-Model Construction*

We fit an analytical equation to the function $f_h(\cdot)$ in (16), due to it's ease of use and minimal memory requirements. The general equation is fixed for all the circuits. This *works* because, even though the function $f_h(\cdot)$ is non-linear, it turns out that in practice it is "not too non-linear" to defy fitting, and a general polynomial template turns out to be sufficient. Moreover, we generate two different analytical equations, one for estimating the zero-delay energy-per-cycle and another for estimating the real-delay energy-per-cycle.

Now we will describe the choice of the polynomial function for estimating real-delay and zero-delay energy-per-cycle.

2.2.1 Macro-Model for Real-Delay Energy-Per-Cycle

One would like to choose the lowest order polynomial equation that works well. One option is the linear function:

$$\begin{aligned} \hat{E}_h\left(\mathbf{x}_1, \mathbf{x}_2\right) = c_0(h) + c_1(h)\mathcal{N}_{s1}^z\left(\mathbf{x}_1, \mathbf{x}_2\right) \\ + c_2(h)\mathcal{N}_{s2}^z\left(\mathbf{x}_1, \mathbf{x}_2\right) \\ + c_3(h)\mathcal{N}_{s3}^z\left(\mathbf{x}_1, \mathbf{x}_2\right) \end{aligned} \tag{17}$$

where the coefficients $c_i(h), i = 0,1,2,3$ are unknown and are to be determined during the characterization using RLS [18,21]. In RLS, the coefficients $c_i(h)$ are determined such that they minimize the following error term:

$$e = \sum_{j=1}^{P} \left(E\left(\mathbf{x}_1, \mathbf{x}_2\right) - \hat{E}_h\left(\mathbf{x}_1, \mathbf{x}_2\right)\right)^2 \tag{18}$$

where P is the number of input vector pairs used for fitting and $E\left(\mathbf{x}_1, \mathbf{x}_2\right)$ is obtained using real-delay gate-level simulator [14], which also provides zero-delay transitions at all levels for the given input vector pair $(\mathbf{x}_1, \mathbf{x}_2)$. One advantage of using RLS is that we do not have to predefine the value of P, because it stops computing the coefficients when some user defined accuracy is reached.

But our goal is to reduce the relative error $(\frac{|E(\mathbf{x}_1,\mathbf{x}_2)-\hat{E}_h(\mathbf{x}_1,\mathbf{x}_2)|}{E(\mathbf{x}_1,\mathbf{x}_2)})$, therefore (18) should be modified to incorporate this. Considering relative error instead of absolute error, (18) becomes:

$$e_m = \sum_{j=1}^{P} \left(\frac{E(\mathbf{x}_1, \mathbf{x}_2) - \hat{E}_h(\mathbf{x}_1, \mathbf{x}_2)}{E(\mathbf{x}_1, \mathbf{x}_2)} \right)^2 \tag{19}$$

where the subscript m stands for *modified.* This simplifies to:

$$e_m = \sum_{j=1}^{P} (1 - R_h(\mathbf{x}_1, \mathbf{x}_2))^2 \tag{20}$$

where $R_h(\mathbf{x}_1, \mathbf{x}_2) = \frac{\hat{E}_h(\mathbf{x}_1,\mathbf{x}_2)}{E(\mathbf{x}_1,\mathbf{x}_2)}$. We rewrite (20) as:

$$e_m = \sum_{j=1}^{P} (y_{new}(\mathbf{x}_1, \mathbf{x}_2) - \hat{y}_{new}(\mathbf{x}_1, \mathbf{x}_2))^2 \tag{21}$$

where $y_{new}(\mathbf{x}_1, \mathbf{x}_2) = 1.0$ and $\hat{y}_{new}(\mathbf{x}_1, \mathbf{x}_2) = R_h(\mathbf{x}_1, \mathbf{x}_2)$. The above equation (21), is a standard RLS [18,21] problem. Since we are monitoring relative error, we have found that by minimizing (21) instead of (18), the accuracy is improved by 5%. Hence while using RLS to estimate the regression variables $c_i(h)$ for each Hamming distance, the modified error criteria (21) is used.

To test the accuracy of the fit, 500 input vector pairs having Hamming distance equal to h, were randomly generated, and $E(\mathbf{x}_1, \mathbf{x}_2)$, $\mathcal{N}^z_{s1}(\mathbf{x}_1, \mathbf{x}_2)$, $\mathcal{N}^z_{s2}(\mathbf{x}_1, \mathbf{x}_2)$, and $\mathcal{N}^z_{s3}(\mathbf{x}_1, \mathbf{x}_2)$ were estimated for every input vector pair $(\mathbf{x}_1, \mathbf{x}_2)$ using [14]. Also, $\hat{E}_h(\mathbf{x}_1, \mathbf{x}_2)$ was estimated for every input vector pair using (17) and the relative error was computed. This procedure was carried out for all Hamming distances $h \in H$. The average of this error over all Hamming distances, is shown for ISCAS-85 [13] benchmark circuits, in Table 3 under the column "$L_{Avg.Error}$", which is calculated as:

$$L_{Avg.Error} = \frac{1}{M}\sum_{i=1}^{M} \frac{1}{P}\sum_{j=1}^{P} \frac{|\hat{E}_{h,j}(\mathbf{x}_1, \mathbf{x}_2) - E_j(\mathbf{x}_1, \mathbf{x}_2)|}{E_j(\mathbf{x}_1, \mathbf{x}_2)} \tag{22}$$

where $P = 500$. It is clear from the table that the linear model works well for some circuits but not for all. Therefore, we have to go to the higher order polynomial function.

Another option is to choose the quadratic function:

$$\begin{aligned}\hat{E}_h(\mathbf{x}_1, \mathbf{x}_2) = {} & c_0(h) + c_1(h)\mathcal{N}^z_{s1}(\mathbf{x}_1, \mathbf{x}_2) + c_2(h)\mathcal{N}^z_{s2}(\mathbf{x}_1, \mathbf{x}_2) + c_3(h)\mathcal{N}^z_{s3}(\mathbf{x}_1, \mathbf{x}_2) \\ & + c_4(h)\mathcal{N}^z_{s1}(\mathbf{x}_1, \mathbf{x}_2)\mathcal{N}^z_{s2}(\mathbf{x}_1, \mathbf{x}_2) + c_5(h)\mathcal{N}^z_{s1}(\mathbf{x}_1, \mathbf{x}_2)\mathcal{N}^z_{s3}(\mathbf{x}_1, \mathbf{x}_2) \\ & + c_6(h)\mathcal{N}^z_{s2}(\mathbf{x}_1, \mathbf{x}_2)\mathcal{N}^z_{s3}(\mathbf{x}_1, \mathbf{x}_2) + c_7(h)\{\mathcal{N}^z_{s1}(\mathbf{x}_1, \mathbf{x}_2)\}^2 \\ & + c_8(h)\{\mathcal{N}^z_{s2}(\mathbf{x}_1, \mathbf{x}_2)\}^2 + c_9(h)\{\mathcal{N}^z_{s3}(\mathbf{x}_1, \mathbf{x}_2)\}^2\end{aligned} \tag{23}$$

The accuracy of the quadratic function was estimated using the same approach as above. The results are shown in Table 3, under the column "$Q_{Avg.Error}$". It can be seen that the average error for all of the circuits is less than 20%. We also investigated the general cubic model. It is not shown here, due to space limitations, as it consists of 20 regression variables. The error for the cubic function is shown in Table 3, under the column marked "$C_{Avg.Error}$". It can be seen from the table that there is little or no improvement, in going from quadratic to cubic model. Therefore, while using RLS we start with a linear model in (16) and change to quadratic model if the desired user accuracy is not satisfied. We do not go beyond quadratic model. Hence the highest order polynomial chosen in (16), while estimating the real-delay energy-per-cycle, was quadratic.

Table 3. Error in the various models while estimating real-delay energy-per-cycle

Circuit	$L_{Avg.Error}$	$Q_{Avg.Error}$	$C_{Avg.Error}$
c499	4.46%	4.43%	5.42%
c880	15.83%	13.47%	14.25%
c1355	12.02%	8.92%	9.36%
c1908	13.61%	12.46%	14.61%
c432	18.56%	15.68%	15.10%
c5315	13.33%	9.65%	9.79%
c2670	21.91%	17.72%	18.09%
c3540	21.69%	19.23%	20.22%
c7552	16.53%	14.64%	15.82%
c6288	21.72%	17.06%	17.86%

2.2.2 Macro-Model for Zero-Delay Energy-Per-Cycle Similar experiments, as that for real-delay, were carried out for zero-delay. It was found that the linear function is "good enough". Hence for estimating zero-delay energy-per-cycle, the macromodel is:

$$\begin{aligned}\hat{E}_h^{zd}(\mathbf{x}_1, \mathbf{x}_2) = c_0(h) + c_1(h)\mathcal{N}_{s1}^z(\mathbf{x}_1, \mathbf{x}_2) \\ + c_2(h)\mathcal{N}_{s2}^z(\mathbf{x}_1, \mathbf{x}_2) \\ + c_3(h)\mathcal{N}_{s3}^z(\mathbf{x}_1, \mathbf{x}_2)\end{aligned} \tag{24}$$

where the superscript zd signifies zero-delay energy-per-cycle. Therefore, highest order polynomial chosen for RLS while estimating zero-delay energy is linear.

3. Extension to Peak-Current-Per-Cycle Estimation

So far, we have presented a macromodeling approach for estimating energy-per-cycle. But in some applications information about the peak-current drawn from the logic circuit in every clock cycle may be desired. The prominent application of peak-current is the analysis of IR-drop problem and the design of the power grid. In order to analysis these problems at high-level of abstraction, we need macromodels which can provide peak-current for every input vector pair $(\mathbf{x}_1, \mathbf{x}_2)$, which we refer to as peak-current-per-cycle. In this section we will extend our energy-per-cycle macromodel to estimate peak-current-per-cycle.

Peak-current-per-cycle ($I_p(\mathbf{x}_1, \mathbf{x}_2)$) is defined as the maximum current drawn from the logic circuit for the input vector pair $(\mathbf{x}_1, \mathbf{x}_2)$. Estimating $I_p(\mathbf{x}_1, \mathbf{x}_2)$ is a difficult task as apart from depending upon the logic circuit, it depends upon the rise/fall times of the waveforms at the primary inputs [22]. Furthermore, it depends upon how many gates are switching simultaneously and what kind of transition they are undergoing [22]. Since, we want to estimate $I_p(\mathbf{x}_1, \mathbf{x}_2)$ at RTL, and due to the dependence of peak-current on many factors, some assumptions are to be made in-order to make macromodel at RTL. We assume that all the primary inputs switch simultaneously and the waveforms applied at the primary inputs are ideal (step input), i.e., they have no rise/fall times. This is not a very crude assumption as some might think, because at high-level it is difficult to find rise/fall times of the waveforms as delay values of different gates are not available.

As before, for making the macromodel for peak-current-per-cycle we target the combinational parts of a given circuit block, for reasons as given while making the macromodel for energy-per-cycle. Therefore, the goal of macromodeling is to find a function $\hat{I}_p(\mathbf{x}_1, \mathbf{x}_2)$ which would be a good approximation to $I_p(\mathbf{x}_1, \mathbf{x}_2)$.

Apart from many factors, prominent factors on which $I_p(\mathbf{x}_1, \mathbf{x}_2)$ depends are the switching activity of the gates and how many of them are switching simultaneously [22]. This is due the fact that whenever output node of a gate makes a transition, it either draws the current from the supply voltage or supplies it to the ground and whenever the gates switch simultaneously the current waveforms get added and hence the current, which may result in peak-current. Therefore, we can model $I_p(\mathbf{x}_1, \mathbf{x}_2)$ as some function of number of transitions at every node:

$$\hat{I}_p(\mathbf{x}_1, \mathbf{x}_2) = g\left(n_1(\mathbf{x}_1, \mathbf{x}_2), n_2(\mathbf{x}_1, \mathbf{x}_2), \cdots, n_N(\mathbf{x}_1, \mathbf{x}_2)\right) \tag{25}$$

where g is some unknown function, $n_i(\mathbf{x}_1, \mathbf{x}_2)$ is the number of transitions at node i due to the input vector pair $(\mathbf{x}_1, \mathbf{x}_2)$, and N is the number of nodes. Note that, while choosing the variables, we did not consider the direction of the transition, as it leads to a more complex model (requiring more variables). Later, in the results section, we will show that, even though we did not consider the direction of transitions, the average error, for all of the circuits that we considered, was less than 20%.

By combining the transition of the nodes at the same level, (25) is given by:

$$\hat{I}_p(\mathbf{x}_1, \mathbf{x}_2) = r\left(\mathcal{N}_1(\mathbf{x}_1, \mathbf{x}_2), \mathcal{N}_2(\mathbf{x}_1, \mathbf{x}_2), \cdots, \mathcal{N}_K(\mathbf{x}_1, \mathbf{x}_2)\right) \tag{26}$$

where r is the new unknown function and K is the number of levels. Determining the value of $\mathcal{N}_i(\mathbf{x}_1, \mathbf{x}_2)$ requires real-delay simulation of the whole circuit, which is prohibitive at RTL. Another possibility would be to construct the model as the function of zero-delay transitions at every level. Un-

$$\hat{I}_p(\mathbf{x}_1, \mathbf{x}_2) = s\left(\mathcal{N}_1^z(\mathbf{x}_1, \mathbf{x}_2), \mathcal{N}_2^z(\mathbf{x}_1, \mathbf{x}_2), \cdots, \mathcal{N}_K^z(\mathbf{x}_1, \mathbf{x}_2)\right) \tag{27}$$

where s is another new unknown function. For the reasons as given in the case of energy-per-cycle, we will have different model for each hamming distance. With this modification, the model of (27)

$$\hat{I}_p(h; \mathbf{x}_1, \mathbf{x}_2) = s_h\left(\mathcal{N}_1^z(\mathbf{x}_1, \mathbf{x}_2), \mathcal{N}_2^z(\mathbf{x}_1, \mathbf{x}_2), \cdots, \mathcal{N}_K^z(\mathbf{x}_1, \mathbf{x}_2)\right) \tag{28}$$

Since this requires performing a functional simulation of the whole circuit, we choose three levels inside the circuit, using stepwise regression procedure, as explained in section 2.1. Denoting $ps1, ps2$, and $ps3$ to be three selected levels, the macromodel (28) becomes:

$$\hat{I}_p(h; \mathbf{x}_1, \mathbf{x}_2) = \mathcal{F}_h\left(\mathcal{N}_{ps1}^z(\mathbf{x}_1, \mathbf{x}_2), \mathcal{N}_{ps2}^z(\mathbf{x}_1, \mathbf{x}_2), \mathcal{N}_{ps3}^z(\mathbf{x}_1, \mathbf{x}_2)\right) \tag{29}$$

Notice that the levels $s1, s2, s3$ chosen in (23) are different from the three selected levels of (29). Again, the type of polynomial function to be fitted to the function $\mathcal{F}(\cdot)$ is determined by using the procedure described in section 2.2. We found that quadratic function works well for all the circuits that we tested. The only difference in characterization of the function $\mathcal{F}(\cdot)$ from that of $f(\cdot)$ is that now we do transistor-level (SPICE) simulation, to find $I_p(\mathbf{x}_1, \mathbf{x}_2)$, instead of doing gate-level simulation.

4. CHARACTERIZATION FLOW

Once we have chosen the functions $f_h(\cdot)$, and $\mathcal{F}(\cdot)$ the macromodels are complete for estimating both energy-per-cycle (real-delay and zero-delay) and peak-current-per-cycle. Now we will explain the characterization flow for constructing and using the macromodels.

The complete characterization flow for constructing the macromodel is as follows:

Step 1. Choose the three levels, using the approach described in section 2.1. Store their Boolean descriptions as a function of primary inputs, in the form of Boolean equations.

Step 2. Find the polynomial model type and the regression coefficients using RLS [18,21], for all Hamming distances.

Step 3. Store the analytical equations for using at RTL.

Now a word about the complexity of our approach. For building our macromodel suppose we have to perform W RLS iterations and W real-delay simulation (gate-level or transistor-level) for each Hamming distance. Note that for every RLS iteration we have to perform one real-delay simulation. Assuming the cost of energy and peak-current estimation for a input vector pair is T_1 and T_2 respectively and that for one RLS iteration is T_3, the total cost of our approach for energy-per-cycle (peak-current-per-cycle) is given by:

$$Cost = WMT_1(T_2) + WMT_3 \qquad (30)$$

which is linear in M. In other words the complexity of our approach is $\mathcal{O}(WM)$. Here W is really a constant number and we found that for most of the circuits W was less than 1000.

Our macromodel is also easy to use and the flow for using the macromodel is given as:

Step 1. For a given input vector pair $(\mathbf{x}_1, \mathbf{x}_2)$, perform fast functional simulation to determine zero-delay transitions at the three selected levels. Note that the user does not has to perform the zero-delay simulation of the whole circuit. The user has to only perform the functional simulation of the boolean equations derived at the nodes of the three levels.

Step 2. Substitute the values determined in step 1, in (23) and (24) for estimating real-delay and zero-delay energy-per-cycle and in (29) for estimating peak-current-per-cycle corresponding to Hamming distance h.

In next section we will demonstrate the accuracy of our macromodeling approaches.

5. RESULTS

First, we will present the results for estimating energy-per-cycle and then for estimating peak-current-per-cycle.

5.1 *Energy-Per-Cycle*

We constructed macromodels for a number of circuits, using our approach as described in section 4. In order to test the accuracy of our approach, we randomly generated around 100,000 input vector pairs. Let us describe how these vectors were generated randomly. Firstly, for each Hamming distance we choose number of zero-to-one and one-to-zero transitions randomly, such that their sum is equal to Hamming distance. Secondly, we choose number of zero-to-zero and one-to-one transitions randomly. Finally, we randomly choose the primary inputs which will have zero-to-one, one-to-zero, one-to-one and zero-to-zero transitions. With these three degrees of randomness added, it is highly unlikely that characterization vectors and testing vectors will be same. Therefore, by high degree of randomness in vector generation we ensure that the characterization vectors and testing vectors are different.

Energy-per-cycle $(\hat{E}_i(\mathbf{x}_1, \mathbf{x}_2))$ was estimated, for every input pair $(\mathbf{x}_1, \mathbf{x}_2)$, using the flow described in section 4. Energy-per-cycle $(E_i(\mathbf{x}_1, \mathbf{x}_2))$ was also estimated using [14]. Table 4 (under the column "Real-delay) shows the average error, for the ISCAS-85 [13] benchmark circuits, a 16-bit ripple carry adder (Adder 16), and a 10×10-bit Baugh-Wooley (BW10) multiplier, under the column "$CE_{Avg.Error}$". This error is calculated as:

$$CE_{Avg.Error} = \frac{1}{P} \sum_{i=1}^{P} \frac{|E_i(\mathbf{x}_1, \mathbf{x}_2) - \hat{E}_i(\mathbf{x}_1, \mathbf{x}_2)|}{E_i(\mathbf{x}_1, \mathbf{x}_2)} \qquad (31)$$

where $P = 100,000$ is the number of test points. It is clear from the table, that the error is less than 20% for all the circuits, that we tested. Moreover, column "AE_{Error}", shows the relative error, while estimating the average energy, which is computed as:

$$AE_{Error} = \frac{|\sum_{i=1}^{P} E_i(\mathbf{x}_1, \mathbf{x}_2) - \sum_{i=1}^{P} \hat{E}_i(\mathbf{x}_1, \mathbf{x}_2)|}{\sum_{i=1}^{P} E_i(\mathbf{x}_1, \mathbf{x}_2)} \quad (32)$$

where $P = 100,000$. Again, the error is less than 10% for all the circuits. In Table 4, the columns marked "#I", "#O", and "#L", show the number of inputs, number of outputs and number of levels in the circuit respectively. Also, shown in Table 4 is the time taken to construct the macromodel. The execution times are on a SUN Ultra Sparc 1 with 64MB of RAM. The longest time is taken by c7552 which has the highest number of primary inputs. It took only a couple of hours to build the macromodel for most of the circuits.

Similar experiments were carried out for zero-delay energy-per-cycle and the results are also shown in Table 4, under the column "Zero-delay". The average error in estimating energy-per-cycle and average power is less than 15% and 5% respectively, for all the circuits. Note that the execution times are the same as that for real-delay energy-per-cycle macromodel, because both the real-delay and zero-delay energy-per-cycle macromodels were constructed simultaneously.

Table 4. Error in the approach while estimating real-delay and zero-delay energy

Circuit	#I	#O	#L	Real-delay		Zero-delay		Time (in hours)
				$CE_{Avg.Error}$	AE_{Error}	$CE_{Avg.Error}$	AE_{Error}	
c499	41	32	11	4.41%	1.01%	4.10%	0.58%	1.56
c880	60	26	24	13.43%	4.14%	11.89%	1.94%	3.99
c1355	41	32	24	9.15%	1.87%	5.77%	0.19%	2.83
c1908	33	25	40	12.45%	7.6%	8.96%	1.02%	2.41
c432	36	7	17	15.69%	7.30%	13.41%	5.03%	1.75
c5315	178	123	49	10.33%	1.73%	6.40%	0.33%	8.72
c2670	157	140	32	17.77%	5.83%	9.54%	1.42%	8.04
c3540	50	22	47	19.32%	6.53%	11.04%	0.99%	2.69
c7552	207	108	43	14.53%	4.38%	6.45%	0.3%	11.73
c6288	32	32	124	17.18%	3.43%	7.58%	0.7%	4.59
Adder16	32	16	33	11.64%	2.65%	9.79%	0.86%	2.21
BW10	20	20	50	14.04%	3.19%	9.17%	1.19%	1.57

Shown in Fig. 4 are various error measures for c1908, while estimating the real-delay energy-per-cycle. In Fig. 4a, we show the relative absolute error ($\frac{|E(\mathbf{x}_1,\mathbf{x}_2) - \hat{E}_h(\mathbf{x}_1,\mathbf{x}_2)|}{E(\mathbf{x}_1,\mathbf{x}_2)}$), averaged over 1000 input vector pairs generated randomly for each Hamming distance, and the standard deviation of this error. It can be seen from the figure that the error decreases as the Hamming distance increases. For low Hamming distances, the error is high as the energy-per-cycle values are very small and a small deviation from the actual value leads to a very large relative error. This is demonstrated in Fig. 4b, where we show, the absolute error ($|E(\mathbf{x}_1,\mathbf{x}_2) - \hat{E}(\mathbf{x}_1,\mathbf{x}_2)|$), averaged over 1000 input vector pairs generated randomly for each Hamming distance, and standard deviation of this error. It is apparent from this figure that even though the relative error was high for small Hamming distances, the absolute error is actually low. Also, the absolute error levels off for high Hamming distances. Fig. 4c shows the energy, averaged over 1000 input vector pairs generated randomly for each Hamming distance, which supports the behavior of errors in Figs. 4a, and 4b. Finally, Fig. 4d shows the error histogram while estimating the energy-per-cycle for 100,000 input vector pairs. The error distribution is centered around zero, and the bulk of the distribution is in a narrow region around zero, but the tails (a very small fraction of cases) are further out than one would like. Further work may help to narrow this distribution. In any case, the power of this approach becomes clear when one considers Fig. 5, which was generated by first applying a low-activity vector sequence, followed by a high activity sequence, as was done for Fig. 3. Fig. 5 shows the real-delay and zero-delay transient energy waveforms respectively, for c1908 and c5315. This behavior is similar to what was shown in Fig. 3 for the golden model. This shows that the model is very useful for tracking changes in the power dissipation over time, and it has no lag time, so that it reacts immediately to a change in the characteristics of the input stream.

Finally, the energy-per-cycle macromodels can be used for estimating the average energy over m input vector pairs $\{(\mathbf{x}_1, \mathbf{x}_2), (\mathbf{x}_2, \mathbf{x}_3), \cdots, (\mathbf{x}_m, \mathbf{x}_{m+1})\}$. The actual and predicted energy, averaged over m input vector pairs, are given by the following expressions:

$$E_m = \frac{1}{m} \sum_{i=1}^{m} E\left(\mathbf{x}_i, \mathbf{x}_{i+1}\right) \tag{33}$$

$$\hat{E}_m = \frac{1}{m} \sum_{i=1}^{m} \hat{E}_h\left(\mathbf{x}_i, \mathbf{x}_{i+1}\right) \tag{34}$$

Shown in the Figs. 6a, 6b, and 6c are the scatter plots of the moving average real-delay energy, for $m = 1$, $m = 5$ and $m = 15$ respectively, for c880, one of the ISCAS-85 [13] benchmark circuit. The number of data points in each plot is 10,000. As the value of m increases, the accuracy in the estimation increases.

5.2 *Peak-Current-Per-Cycle*

We constructed macromodels for a number of circuits, using our approach as described in section 4. In order to test the accuracy of our approach, we randomly generated around 10,000 input vector pairs. Peak-current-per-cycle ($\hat{I}_p\left(i; \mathbf{x}_1, \mathbf{x}_2\right)$) was estimated, for every input pair $(\mathbf{x}_1, \mathbf{x}_2)$, using the flow described in section 4. Peak-current-per-cycle ($I_p\left(i; \mathbf{x}_1, \mathbf{x}_2\right)$) was also estimated using SPICE, for three ISCAS-85 [13] and some of the MCNC benchmark circuits. Note that constructing macromodels for rest of the ISCAS-85 [13] circuits were not feasible due to the long run-time of SPICE. Shown in Fig. 7 is the combined scatter plot. It can be seen from the figure that the fit is good and it is indeed possible to estimate peak-current-per-cycle at RTL. Table 5 shows the average error which is calculated as:

$$Avg.Error = \frac{1}{P} \sum_{i=1}^{P} \frac{\left|I_p\left(i; \mathbf{x}_1, \mathbf{x}_2\right) - \hat{I}_p\left(i; \mathbf{x}_1, \mathbf{x}_2\right)\right|}{I_p\left(i; \mathbf{x}_1, \mathbf{x}_2\right)} \tag{35}$$

where $P = 10,000$ is the number of test points. It is clear from the table, that the error is less than 20% for all the circuits, that we tested. In table 5, the columns marked "#I", "#O", "#L", and "#G" show the number of inputs, number of outputs, number of levels, and number of gates in the circuit respectively. Also, shown in Table 5 is the time taken to construct the macromodel. The execution times are on a SUN Ultra Sparc 1 with 64MB of RAM. It can be seen from the table that it took days to build macromodel for the circuits with modest size of netlist, due to the long run-time of SPICE. In practice, one can use faster transistor-level simulators, such as PowerMill or iRSIM. In any case, that this is only a one-time cost.

Shown in Fig. 8 are various error measures for c880, while estimating the peak-current-per-cycle. This figure is similar to the Fig. 4. In Fig. 8a, we show the relative absolute error ($\frac{\left|I_p(\mathbf{x}_1,\mathbf{x}_2) - \hat{I}_p(h;\mathbf{x}_1,\mathbf{x}_2)\right|}{I_p(\mathbf{x}_1,\mathbf{x}_2)}$), averaged over 100 input vector pairs generated randomly for each Hamming distance, and the standard deviation of this error. It can be seen from the figure that the error decreases as the Hamming distance increases. For low

Hamming distances, the error is high as the peak-current-per-cycle values are very small and a small deviation from the actual value leads to a very large relative error. This is demonstrated in Fig. 8b, where we show, the absolute error ($|I_p(\mathbf{x}_1, \mathbf{x}_2) - \hat{I}_p(\mathbf{x}_1, \mathbf{x}_2)|$), averaged over 100 input vector pairs generated randomly for each Hamming distance, and standard deviation of this error. It is apparent from this figure that even though the relative error was high for small Hamming distances, the absolute error is actually low. Fig. 8c shows the peak-current, averaged over 100 input vector pairs generated randomly for each Hamming distance, which supports the behavior of errors in Figs. 8a, and 8b. Finally, Fig. 8d shows the error histogram while estimating the peak-current-per-cycle for 10,000 input vector pairs. The error distribution is centered around zero, and the bulk of the distribution is in a narrow region around zero, but the tails (a very small fraction of cases) are further out than one would like. But, this much error is acceptable considering the fact that we are estimating peak-current at RTL. However, future work may lead to the improvement in the macromodel.

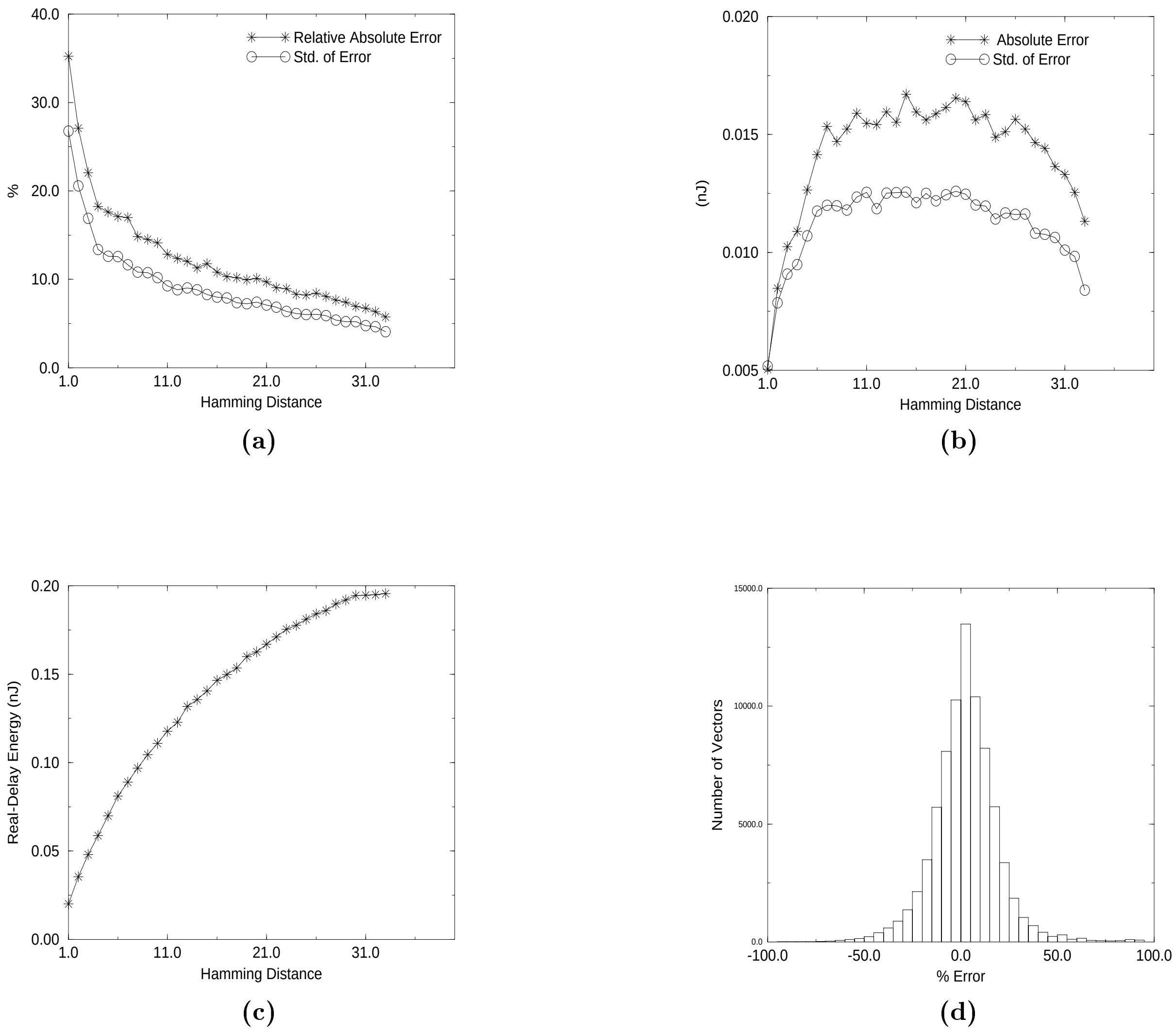
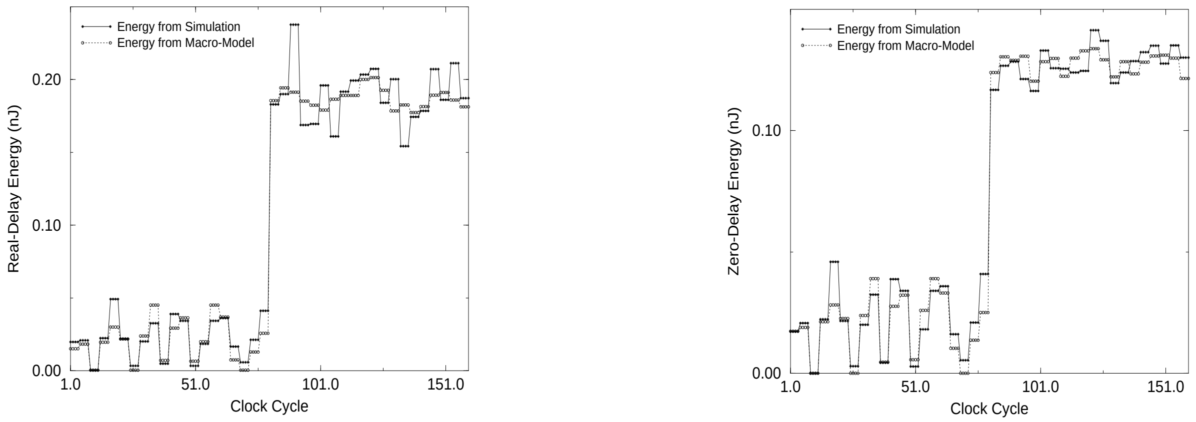
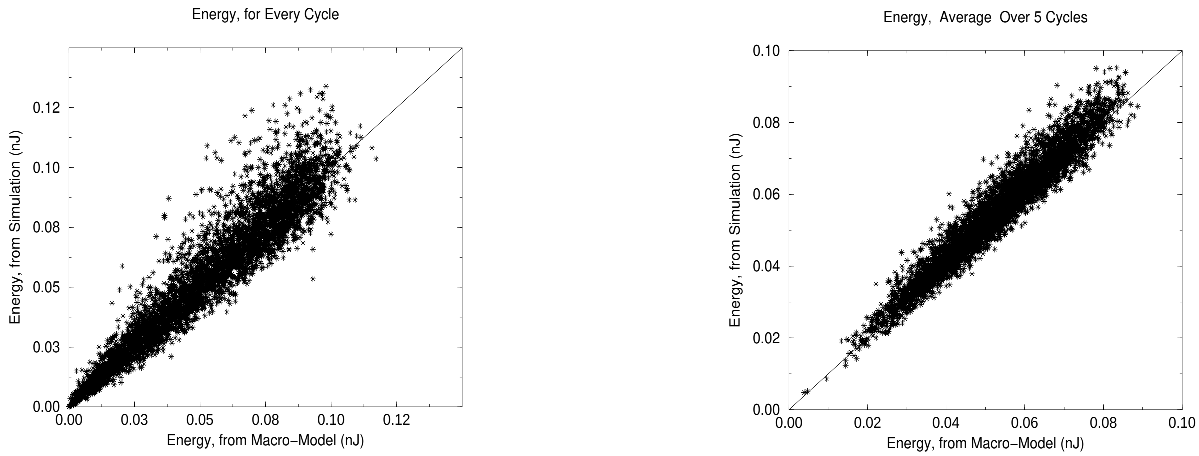


Figure 4. Showing various error measures for c1908.

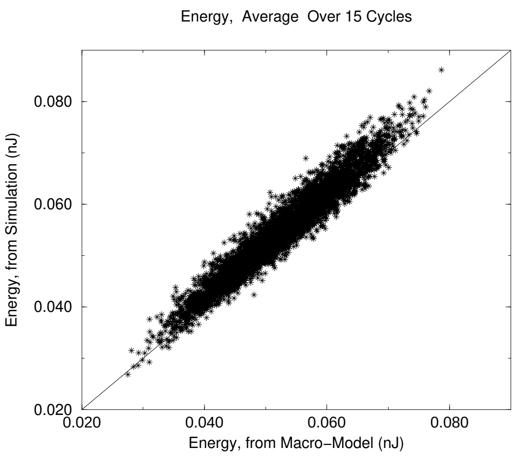


(a) **(b)**

Figure 5. Showing transient energy waveforms for c1908.



(a) **(b)**



(c)

Figure 6. Scatter plots of moving average energy for c880.

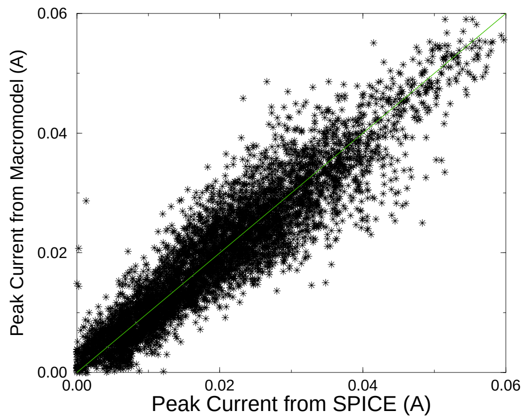


Figure 7. Scatter plot for ISCAS-85 and MCNC benchmark circuits.

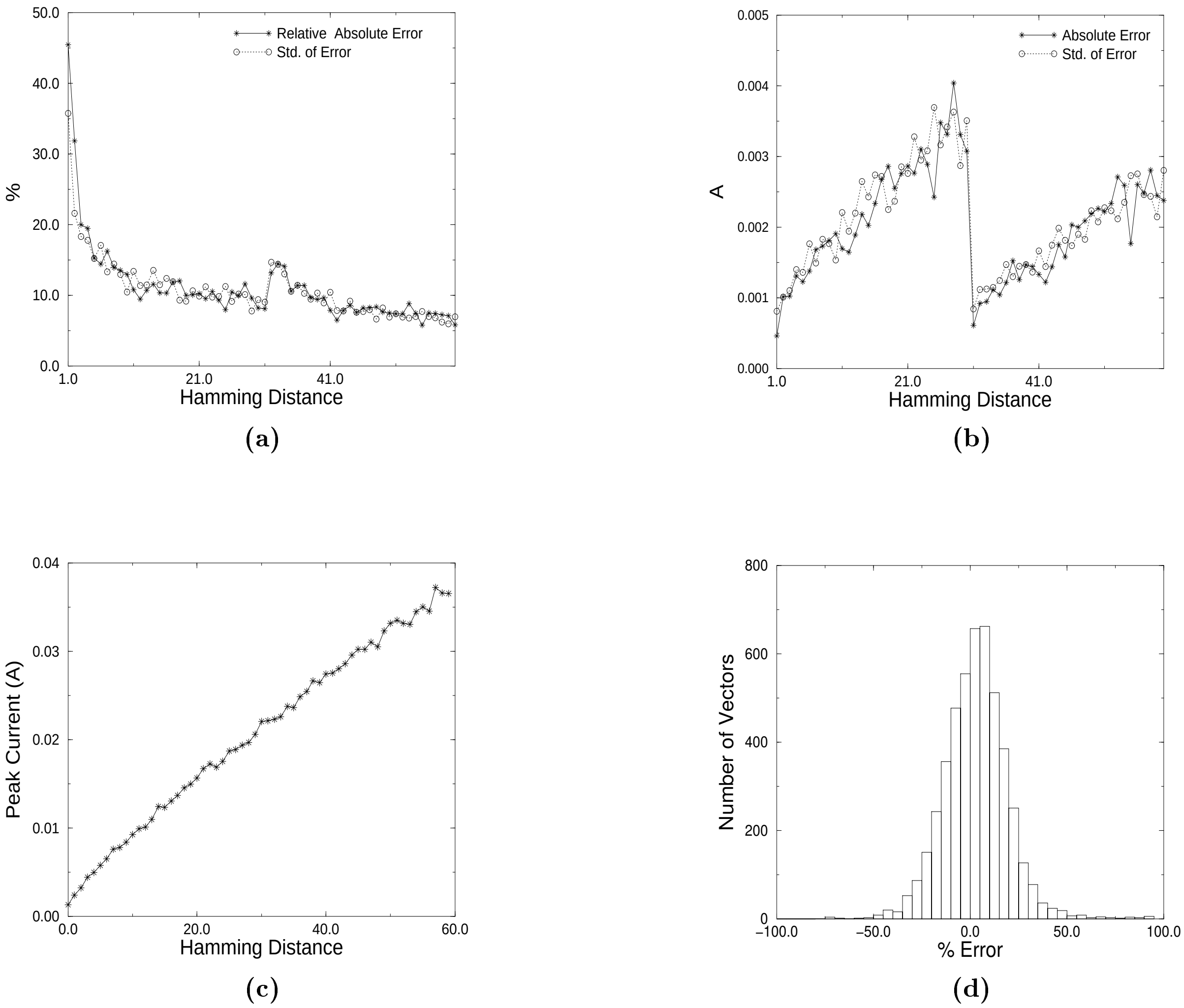


Figure 8. Showing various error measures for c880 while estimating peak-current-per-cycle.

Table 5. Error in the approach while estimating peak-current-per-cycle

Circuit	#I	#O	#L	#G	Avg. Error	Time
c499	41	32	11	202	6.95%	3.32days
c880	60	26	24	383	15.51%	4.51days
c432	36	7	17	160	14.77%	1.58days
alu2	10	4	40	368	13.40%	21.94hrs
cu	14	8	12	63	17.91%	3.65hrs
f51m	8	1	33	84	14.89%	3.11hrs
mux	21	1	10	47	19.82%	5.07hrs
parity	16	1	12	75	9.21%	4.17hrs
pcler8	27	17	11	101	17.06%	10.5hrs
random8	8	1	16	158	13.75%	5.77hrs
sct	19	7	41	83	14.87%	5.81hrs
x2	10	3	14	50	17.92%	2.5hrs
z4ml	7	1	20	44	15.30%	1.36hrs
vdao	17	27	15	341	15.57%	2.95days

6. Conclusion

We presented a novel macromodeling approach for estimating the energy and peak-current for every input vector pair (energy-per-cycle and peak-current-per-cycle). This capability is useful in order to study the changes over time in the power dissipation of logic circuits, with applications in power grid analysis (IR-drop, noise, inductive kick), thermal analysis, etc. Some key features of this technique are: 1) the model is compact (linear in the number of inputs), 2) it can be used for *any* input sequence and does not require tuning or the use of a training set, and 3) the characterization is automatic and requires no user intervention. The discussion has focused on combinational circuits, mainly because they represent the most difficult challenge, from a modeling standpoint. It is trivial to combine our model with cell level models of the registers or flip-flops in order to model the energy-per-cycle of large sequential systems.

The macromodel is based on classifying vector pairs on the basis of their Hamming distances and using equation-based macromodels for every Hamming distance. The equations are in terms of three variables, namely the transition counts resulting from evaluation of Boolean functions at three internal logic levels. The equations contain coefficients that are determined using least squares fitting, during an automatic characterization process.

The average error in estimating peak-current-per-cycle was under 20%. Furthermore, the average error while estimating the energy-per-cycle was found to be under 20% and if one ignores glitches, the average error becomes under 15%. The energy-per-cycle model can also be used to measure the long-term average power, with an observed error of under 10%, on average and if glitches are ignored, this becomes 5%. But the power of this technique becomes evident in figure like Fig. 5 which shows that the method is very good at tracking changes in the power dissipation over time, with a zero lag time.

References

[1] F. N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on VLSI Systems*, pp. 446-455, Dec.

1994.

[2] P. Landman, *High-level power estimation*, "International Symposium on Low Power Electronics and Design," pp. 29–35, Monterey, CA, August 12–14, 1996.

[3] M. Nemani and F. N. Najm, "Towards a High-Level Power Estimation Capability," *IEEE Transactions on CAD*, vol. 15 pp. 588-598, June 1996.

[4] D. Marculescu, R. Marculescu and M. Pedram, "Information Theoretic Measures of Energy Consumption at Register Transfer Level," *IEEE Transactions on CAD*, vol. 15 pp. 599-610, June 1996.

[5] S. R. Powell and P. M. Chau, " Estimating Power Dissipation of VLSI signal Processing Chips: The PFA technique," *VLSI Signal Processing IV*, pp. 250-259, 1990.

[6] P. E. Landman and J. M. Rabaey, "Architectural Power Analysis: The Dual Bit Type Method," *IEEE Transactions on VLSI*, vol. 3 pp. 173-187 June 1995.

[7] H. Mehta, R. M. Owens and M. J. Irwin, "Energy Characterization based on Clustering," *33rd ACM/IEEE Design Automation Conference*, pp. 702-707, June 1996.

[8] A. Raghunathan, S. Dey and N. K. Jha, "Register-Transfer Level Estimation Techniques for Switching Activity and Power Consumption," *IEEE International Conference on Computer-Aided Design*, pp. 158-165, November 1996.

[9] Q. Qiu, Q. Wu, Chih-S. Ding, and M. Pedram, "Cycle-accurate macro-models for RT-level power analysis," *IEEE Trans. VLSI Systems*, vol. 6, pp. 520-528, no. 4, December 1998.

[10] A. Bogliolo and L. Benini, "Node Sampling: a Robust RTL Power Modeling Approach," *IEEE International Conference on Computer-Aided Design*, pp. 461-467, November 1998.

[11] F. N. Najm, "Transition Density: A New Measure of Activity in Digital Circuits," *IEEE Trans. on CAD*, vol. 12, pp. 310-323, Feb. 1993.

[12] S. Gupta and F. N. Najm, "Power Macromodeling for High Level Power Estimation,"*IEEE Transactions on VLSI Systems*, vol. 8, no. 1, pp. 18-29, Feb. 2000.

[13] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran," *IEEE International Symposium on Circuits and Systems*, pp. 695-698, June 1985.

[14] M. Xakellis and F. N. Najm, "Statistical Estimation of the Switching Activity in Digital Circuits," *31st ACM/IEEE Design Automation Conference*, pp. 728-733, June 1994.

[15] G. Seber, *Linear Regression Analysis.* New York: John Wiley & Sons, 1977.

[16] Z. Chen and K. Roy, "A Power Macromodeling Technique based on Power Sensitivity," *35th ACM/IEEE Design Automation Conference*, pp. 678-683, June 1998.

[17] A. Bogliolo, L. Benini, and G. D. Micheli, "Characterization-Free Behavioral Power Modeling," *Design Automation and Test in Europe (DATE)*, pp. 767-773, Feb. 1998.

[18] S. Gupta and F. N. Najm, "Analytical Model for High Level Power Modeling of Combinational and Sequential Circuit," in *IEEE Alessandro Volta Memorial International Workshop on Low Power Design*, Como, Italy, March 4-5, 1999.

[19] Z. Chen and K. Roy, "Estimation of Power Sensitivity in Sequential Circuits with Power Macromodeling Application," *IEEE International Conference on Computer-Aided Design*, pp. 468-472, November 1998.

[20] S. Even, *Graph Algorithms.* Rockville, MD: Computer Science Press, 1979.

[21] L. Ljung, *System Identification: theory for the user.* Engelwood Cliffs, NJ: Prentice Hall, 1987.

[22] H. Kriplani, F. N. Najm, and I. Hajj, "Pattern independent maximum current estimation in power and ground buses of CMOS VLSI circuits: algorithms, signal correlations, and their resolution,"*IEEE Transactions on Computer-Aided Design*, vol. 14, no. 8, pp. 998-1012, August 1995.

[23] S. Neter and W. Wasserman, *Applied Linear Statistical Models.* Homewood, IL: Richard D. Irwin, Inc., 1974.

[24] S. Gupta and F. N. Najm, "Energy-per-cycle estimation at RTL," *IEEE International Symposium on Low Power Electronics and Design*, San Diego, CA, pp. 121-126, Aug. 16-17, 1999