Energy-Per-Cycle Estimation at RTL†

Subodh Gupta and Farid N. Najm

ECE Dept. and Coordinated Science Lab.
University of Illinois at Urbana-Champaign

Abstract – We present a novel macromodeling technique for estimating the energy dissipated in a logic circuit for every input vector pair (we call this the *energy-per-cycle*). The macromodel is 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. This energy-per-cycle macromodel provides a transient energy waveform, and can also be used to estimate the moving average energy over any time window. This approach has been implemented and models have been built and tested for many circuits. The average error observed in estimating the energy-per-cycle is under 20%. The model can also be used to measure the long-term average power, with an observed error of under 10% on average.

1. Introduction

With the advent of portable and high-density microelectronic devices, the power dissipation of very large scale integrated (VLSI) circuits has become a critical concern. 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. In order to avoid problems associated with excessive power consumption, there is a need for CAD tools to help in estimating the power consumption of VLSI circuits, at various levels of abstraction.

In order to reduce the number of design cycles, power estimation tools are required that can estimate the power consumption at a high level of abstraction. In response to this need, a number of high-level power estimation techniques have been recently proposed which can be grouped into top-down and bottom-up techniques. In the top-down techniques [1], a combinational circuit is specified only as a Boolean function, with no information on the circuit structure, number of gates/nodes, etc. In contrast, bottom-up methods [2–8] 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.

†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, customization funds from Texas Instruments Inc.).

The methods in [2–6] 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) is to build macromodel that gives the energy dissipated in the circuit due to a given vector pair, effectively the energy-per-cycle. This is a simpler problem, for which some solutions [7, 8] have been proposed.

The method in [7] characterizes the power dissipation of circuits based on input transitions. But it is not clear how efficient the method would be on large circuits. In [8], the authors presented a macromodel for estimating the cycle-by-cycle power at the RTL. Their 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 without running the risk of a combinatorial explosion and without requiring a training set to tune the model before it is used. 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 3) inside the circuit. Our equation-based macromodels consist of 10 and 5 coefficients for real-delay and zero-delay respectively. Moreover, this energy-per-cycle macromodel can be used for estimating the average energy over any specified time period.

A key advantage of our approach is that all types of circuits are treated in the same way, i.e., we do not use different model equation types for different circuits. As a result, the method is very easy to use, and requires no user intervention. Indeed, we will present an automatic characterization procedure, by which such equation-based energy-per-cycle macromodels can be built.

This paper is organized as follows. In the next section, we give an overview of our approach. In section 3, we describe the variable selection procedure. In section 4, we discuss the construction of the equation-based macromodel. In section 5, we give the characterization flow of our method. In section 6, we present results and some applications of our macromodels and finally in section 7, we give some conclusions.

2. Energy-per-cycle Estimation

We assume that a circuit block is given that is described at a low level of abstraction (say, at the gate level). We assume that 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 the circuit 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 want 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 may be referred to as a *structural RTL* (Register Transfer Level).

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 a cell-level model for them. The main difficulty lies in modeling the energy-per-cycle for the combinational logic parts so that their energy-per-cycle 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 $n_i(\mathbf{x}_1, \mathbf{x}_2)$ 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:

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

We refer to $E(\mathbf{x}_1, \mathbf{x}_2)$ as the energy-per-cycle. As given, this model seems to ignore the short-circuit power. However, it is common practice to use a modified value of C_i that effectively includes the contribution of the short-circuit power of a logic gate, from a knowledge of its fanout and fanin gates. 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. However, for a circuit with M primary inputs, the total number of possible input vector pairs is 4^M, making this approach practically infeasible for all but the smallest circuits.

Therefore, the goal of macromodeling is to find a function $\hat{E}(\mathbf{x}_1, \mathbf{x}_2)$ 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 3.

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) [6]. We call this step *macromodel construction* and is described in section 4.

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

3. 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 gate whose inputs are either outputs of level 1 gates or are primary inputs is said to have level 2, etc. The levelization algorithm [11] 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*. By grouping the nodes which are at the same level, (1) can be rewritten as:

$$E(\mathbf{x}_1, \mathbf{x}_2) = 0.5 V_{dd}^2 \sum_{i=1}^{K} \sum_{j=1}^{G_i} C_j n_j(\mathbf{x}_1, \mathbf{x}_2) \tag{3}$$

where G_i is number of nodes that are outputs of gates at level i. 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, (3) simplifies to:

$$E(\mathbf{x}_1, \mathbf{x}_2) \approx 0.5 V_{dd}^2 \sum_{i=1}^{K} \mathcal{Q}_i \mathcal{N}_i(\mathbf{x}_1, \mathbf{x}_2) \tag{4}$$

where

$$\mathcal{Q}_i = \frac{1}{G_i} \sum_{j=1}^{G_i} C_j \quad \text{and} \quad \mathcal{N}_i(\mathbf{x}_1, \mathbf{x}_2) = \sum_{j=1}^{G_i} n_j(\mathbf{x}_1, \mathbf{x}_2) \tag{5}$$

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 (4) (x-axis) and that obtained from (3) (y-axis), for c6288, one of the ISCAS-85 [9] 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 [10]. 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 [9] benchmark circuits, which supports the approximation made in (4).

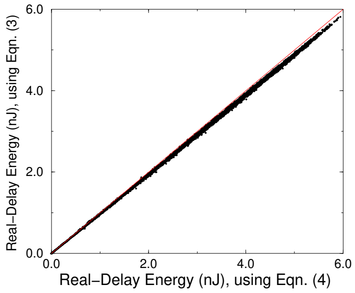


Figure 1. Plot of energy-per-cycle for c6288.

In (4), $\mathcal{Q}_i$ 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 (4) 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:

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

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 [10] 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 (6) and the relative error between the two energy values was computed. Table 1 shows the average of this error, for the ISCAS-85 [9] benchmark circuits, which is computed as:

$$\text{Avg. Error} = \frac{1}{P} \sum_{i=1}^{P} \frac{|E_i(\mathbf{x}_1, \mathbf{x}_2) - \hat{E}_i(\mathbf{x}_1, \mathbf{x}_2)|}{E_i(\mathbf{x}_1, \mathbf{x}_2)} \tag{7}$$

where $P = 100,000$ is the number of input vector pairs.

It is clear from the table that the simple model of (6) is not good enough for estimating the real-delay energy-per-cycle as the glitches are not accounted for in (6). Another possibility is to construct the model as follows:

$$\hat{E}(\mathbf{x}_1, \mathbf{x}_2) = c_0 + \sum_{i=1}^{K} c_i \mathcal{N}_i^z(\mathbf{x}_1, \mathbf{x}_2) \tag{8}$$

where the regression coefficients c_i would be determined using least squares fitting. Note that $\mathcal{Q}_i$ does not appear in (8), as it is contained in the regression coefficient c_i. In fact, we have found that the accuracy of (8) 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 [9] 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.

With this modification, the model of (8) becomes:

$$\hat{E}_h(\mathbf{x}_1, \mathbf{x}_2) = c_0(h) + \sum_{i=1}^{K} c_i(h) \mathcal{N}_i^z(\mathbf{x}_1, \mathbf{x}_2) \tag{9}$$

where the regression coefficients $c_i(h)$ are determined for each Hamming distance using RLS [6].

We call the macromodel of (9), 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 [10], which also provides estimate of $\mathcal{N}_i^z(\mathbf{x}_1, \mathbf{x}_2)$. Energy-per-cycle was also estimated using (9) and the relative error was computed for every input pair. Table 2, shows the average of this error, for ISCAS-85 [9] benchmark circuits, which is computed using (7) with $P = 100,000$. Also, shown in Fig. 3 is the real-delay energy-per-cycle waveform for c1908, an ISCAS-85 [9] 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 1. Error in estimating real-delay energy-per-cycle using macromodel given by (6).

Circuit	Avg.Error	Circuit	Avg.Error
c499	76.5%	c5315	83.5%
c880	104.3%	c2670	104.3%
c1355	81.5%	c3540	103.5%
c1908	43.2%	c7552	86.2%
c432	92.1%	c6288	818.9%

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

Circuit	Avg.Error	Circuit	Avg.Error
c499	3.3%	c5315	8.3%
c880	9.2%	c2670	13.2%
c1355	5.8%	c3540	14.7%
c1908	8.9%	c7552	7.1%
c432	8.2%	c6288	13.3%

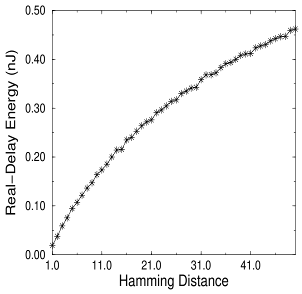


Figure 2. Showing the variation in energy for different Hamming distances, for c3540.

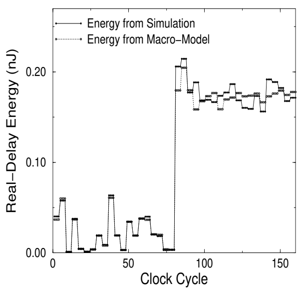


Figure 3. Energy waveform predicted using *golden model* for c1908.

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 (9) 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 (7), over a large number of input vector pairs, was within 20% (see section 6), so that we lose only 5% in accuracy relative to our golden macromodel (9), 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 [12] to find these three levels. Stepwise regression is a well known variable selection method based on F^* *statistics* from regression theory [12].

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 (8).
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 (8), 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 [12], 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:

$$F_k^* = \frac{SSR(v_k)}{MSE(v_k)} \tag{10}$$

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:

$$F_k^* = \frac{SSR(v_k \mid v_i)}{MSE(v_i, v_k)} \tag{11}$$

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:

$$F_i^* = \frac{SSR(v_i \mid v_j)}{MSE(v_j, v_i)} \tag{12}$$

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 (8) 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:

$$\hat{E}_h(\mathbf{x}_1, \mathbf{x}_2) = f_h\left(\mathcal{N}_{s1}^z(\mathbf{x}_1, \mathbf{x}_2), \mathcal{N}_{s2}^z(\mathbf{x}_1, \mathbf{x}_2), \mathcal{N}_{s3}^z(\mathbf{x}_1, \mathbf{x}_2)\right) \tag{13}$$

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

4. Macro-model Construction

In this section, a procedure for determining the order of the polynomial function $f_h(\cdot)$ in (13) is presented. Moreover, we generate two different equations, for estimating the real-delay and zero-delay energy-per-cycle.

4.1 Macro-model for Real-delay Energy

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

$$\hat{E}_h(\mathbf{x}_1, \mathbf{x}_2) = c_0(h) + \sum_{i=1}^{3} c_i(h) \mathcal{N}_{si}^z(\mathbf{x}_1, \mathbf{x}_2) \tag{14}$$

where the coefficients $c_i(h), i = 0, 1, 2, 3$ are unknown and are to be determined during the characterization using RLS [6]. In RLS, the coefficients $c_i(h)$ are determined such that they minimize the following error term:

$$e = \sum_{j=1}^{P} \left(E(\mathbf{x}_1, \mathbf{x}_2) - \hat{E}_h(\mathbf{x}_1, \mathbf{x}_2)\right)^2 \tag{15}$$

where P is the number of input vector pairs used for fitting and $E(\mathbf{x}_1, \mathbf{x}_2)$ is obtained using real-delay gate-level simulator [10], 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 RLS stops computing the coefficients when the user defined accuracy is achieved.

But our goal is to reduce the relative error ($\frac{\left|E(\mathbf{x}_1,\mathbf{x}_2) - \hat{E}_h(\mathbf{x}_1,\mathbf{x}_2)\right|}{E(\mathbf{x}_1,\mathbf{x}_2)}$), therefore (15) should be modified to incorporate this. Considering relative error instead of absolute error, (15) becomes:

$$e_m = \sum_{j=1}^{P} \left(\frac{E(\mathbf{x}_1, \mathbf{x}_2) - \hat{E}_h(\mathbf{x}_1, \mathbf{x}_2)}{E(\mathbf{x}_1, \mathbf{x}_2)}\right)^2 \tag{16}$$

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

$$e_m = \sum_{j=1}^{P} \left(1 - R_h(\mathbf{x}_1, \mathbf{x}_2)\right)^2 \tag{17}$$

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 (17) as:

$$e_m = \sum_{j=1}^{P} \left(y_m(\mathbf{x}_1, \mathbf{x}_2) - \hat{y}_m(\mathbf{x}_1, \mathbf{x}_2)\right)^2 \tag{18}$$

where $y_m(\mathbf{x}_1, \mathbf{x}_2) = 1.0$ and $\hat{y}_m(\mathbf{x}_1, \mathbf{x}_2) = R_h(\mathbf{x}_1, \mathbf{x}_2)$. The above equation (18), is a standard RLS [6] problem. Since we are monitoring relative error, we have found that by minimizing (18) instead of (15), 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 (18) is used.

To test the accuracy of the fit, 500 input vector pairs having Hamming distance equal to h, were randomly generated, and $E(\mathbf{x}_1, \mathbf{x}_2)$, $\mathcal{N}_{s1}^z(\mathbf{x}_1, \mathbf{x}_2)$, $\mathcal{N}_{s2}^z(\mathbf{x}_1, \mathbf{x}_2)$, and $\mathcal{N}_{s3}^z(\mathbf{x}_1, \mathbf{x}_2)$ were estimated for every input vector pair $(\mathbf{x}_1, \mathbf{x}_2)$ using [10]. Also, $\hat{E}_h(\mathbf{x}_1, \mathbf{x}_2)$ was estimated for every input vector pair using (14) 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 [9] benchmark circuits, in Table 3 under the column "$L_{Avg.Error}$", which is calculated as:

$$L_{Avg.Error} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{P}\sum_{j=1}^{P}\frac{|\hat{E}_{h,j}(\mathbf{x}_1, \mathbf{x}_2) - E_j(\mathbf{x}_1, \mathbf{x}_2)|}{E_j(\mathbf{x}_1, \mathbf{x}_2)} \tag{19}$$

where $P = 500$. It is clear from the table that the linear model works well for some circuits but not for all.

Another option is to choose the quadratic function, which will have 10 regression coefficients. It is not shown here due to space limitations. Again 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 again 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 a quadratic to a cubic model. Therefore, we start with a linear model in (13) and change to a quadratic model if the desired user accuracy is not satisfied. We do not go beyond quadratic model.

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

Circuit	$L_{Avg.Error}$	$Q_{Avg.Error}$	$C_{Avg.Error}$	Circuit	$L_{Avg.Error}$	$Q_{Avg.Error}$	$C_{Avg.Error}$
c499	4.5%	4.4%	5.4%	c5315	13.3%	9.7%	9.8%
c880	15.8%	13.5%	14.2%	c2670	21.9%	17.7%	18.1%
c1355	12.1%	8.9%	9.4%	c3540	21.7%	19.2%	20.2%
c1908	13.6%	12.4%	14.6%	c7552	16.5%	14.6%	15.8%
c432	18.5%	15.7%	15.1%	c6288	21.7%	17.1%	17.8%

4.2 Macro-model for Zero-delay Energy

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:

$$\hat{E}_h^{zd}(\mathbf{x}_1, \mathbf{x}_2) = c_0(h) + \sum_{i=1}^{3} c_i(h)\mathcal{N}_{si}^z(\mathbf{x}_1, \mathbf{x}_2) \tag{20}$$

where the superscript zd signifies zero-delay energy-per-cycle.

5. Characterization Flow

Once we have chosen the function $f_h(\cdot)$, the macromodel is complete for estimating both real-delay and zero-delay energy-per-cycle. The complete characterization flow for constructing the macromodel is as follows:

Step 1. Choose the three levels, using the approach described in section 3. 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 [6], for all Hamming distances.

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

Now a word about the complexity of our approach. Every RLS iteration involves a real-delay simulation of the circuit for one input vector pair. If the time required for such a simulation is T, and if W RLS iterations are performed in total, then the time cost of building the macromodel is:

$$Cost = WMT \tag{21}$$

In practice, W for most of the circuits was less than 1000.

Once the macromodel has been built, the flow for *using* it to do power estimation is as follows:

Step 1. For a given input vector pair $(\mathbf{x}_1, \mathbf{x}_2)$, perform fast functional simulation to determine $\mathcal{N}_{s1}^z(\mathbf{x}_1, \mathbf{x}_2)$, $\mathcal{N}_{s2}^z(\mathbf{x}_1, \mathbf{x}_2)$, and $\mathcal{N}_{s3}^z(\mathbf{x}_1, \mathbf{x}_2)$.

Step 2. Substitute the values determined in step 1, in the regression equations (obtained in section 4) for estimating real-delay and zero-delay energy-per-cycle, corresponding to Hamming distance h.

6. Results

In order to test the accuracy of our approach, we randomly generated around 100,000 input vector pairs. 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 5. Energy-per-cycle ($E_i(\mathbf{x}_1, \mathbf{x}_2)$) was also estimated using [10]. Table 4 shows the average error, for the ISCAS-85 [14] benchmark circuits, a 16-bit ripple carry adder (RCA 16), and a 10×10-bit Baugh-Wooley (BW10) multiplier, under "RD" in the column "$CE_{Avg.Error}$". This error is calculated as:

$$CE_{Avg.Error} = \frac{1}{P}\sum_{i=1}^{P}\frac{|E_i(\mathbf{x}_1, \mathbf{x}_2) - \hat{E}_i(\mathbf{x}_1, \mathbf{x}_2)|}{E_i(\mathbf{x}_1, \mathbf{x}_2)} \tag{22}$$

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

$$AE_{Error} = \frac{|\sum_{i=1}^{P} E_i(\mathbf{x}_1, \mathbf{x}_2) - \sum_{i=1}^{P} \hat{E}_i(\mathbf{x}_1, \mathbf{x}_2)|}{\sum_{i=1}^{P} E_i(\mathbf{x}_1, \mathbf{x}_2)} \tag{23}$$

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 (in hours) 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.

Table 4. Error while estimating energy-per-cycle

Circuit	#I	#O	#L	$CE_{Avg.Error}$		AE_{Error}		Time (hrs)
				RD	ZD	RD	ZD	
c499	41	32	11	4.4%	4.1%	1.0%	0.6%	1.56
c880	60	26	24	13.4%	11.9%	4.1%	1.9%	3.99
c1355	41	32	24	9.2%	5.8%	1.9%	0.2%	2.83
c432	36	7	17	15.7%	13.4%	7.3%	5.0%	1.75
c5315	178	123	49	10.3%	6.4%	1.7%	0.3%	8.72
c2670	157	140	32	17.7%	9.5%	5.8%	1.4%	8.04
c3540	50	22	47	19.3%	11.0%	6.5%	1.0%	2.69
c7552	207	108	43	14.5%	6.5%	4.4%	0.3%	11.73
c6288	32	32	124	17.2%	7.6%	3.4%	0.7%	4.59
c1908	33	25	40	12.5%	8.9%	7.6%	1.0%	2.41
RCA16	32	16	33	11.6%	9.8%	2.7%	0.9%	2.21
BW10	20	20	50	14.1%	9.2%	3.2%	1.2%	1.57

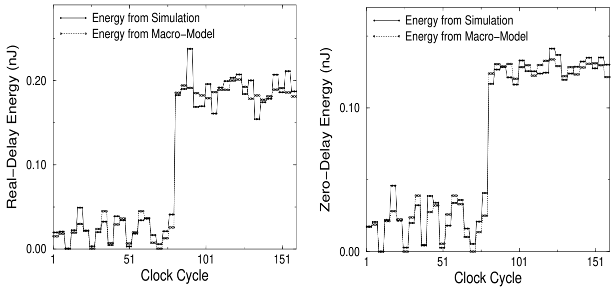
Similar experiments were carried out for zero-delay energy-per-cycle and the results are shown in Table 4, in the columns marked "ZD." The error is less than 15% and 5% respectively, for all the circuits. Note that the execution times are the same as that for real-delay energy-per-cycle macromodel, because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

In any case, the power of this approach becomes clear when one considers Figs. 4 and 5, which were generated by first applying a low-activity vector sequence, followed by a high activity sequence, as was done for Fig. 3. Figs. 4 and 5 show 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:

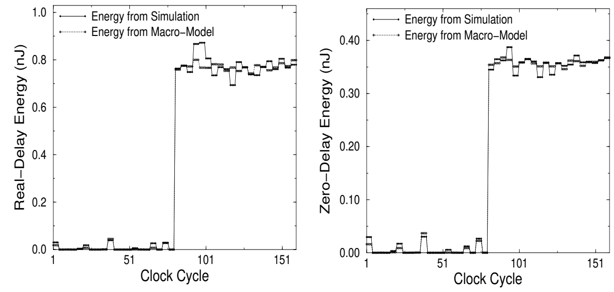
$$E_m = \frac{1}{m} \sum_{i=1}^{m} E\left(\mathbf{x}_i, \mathbf{x}_{i+1}\right) \qquad \hat{E}_m = \frac{1}{m} \sum_{i=1}^{m} \hat{E}_h\left(\mathbf{x}_i, \mathbf{x}_{i+1}\right) \tag{24}$$

Shown in Figs. 6a, and 6b are scatter plots of the moving average real-delay energy, for $m = 1$, and $m = 15$ respectively, for c880, one of the ISCAS-85 [9] 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.



(a) (b)

Figure 4. Transient energy waveforms for c1908.



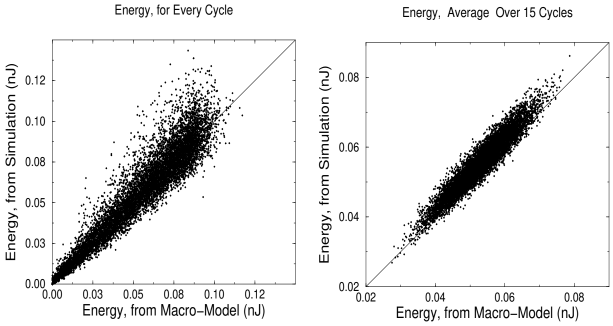
(a) (b)

Figure 5. Transient energy waveforms for c5315.

7. Conclusion

We presented a novel macromodeling approach for estimating the energy for every input vector pair (energy-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. 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 average error while estimating the energy-per-cycle was found to be under 20%. If one ignores glitches, the average error becomes under 15%. The model can also be used to measure the long-term average power, with an observed error of under 10%, on average. If glitches are ignored, this becomes 5%. But the power of this technique becomes evident in figures like Figs. 4 and 5, which show that the method is very good at tracking changes in the power dissipation over time, with a zero lag time.



(a) (b)

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

References

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

[2] 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.

[3] S. Gupta and F. N. Najm, "Power Macromodeling for High Level Power Estimation," *34th ACM/IEEE Design Automation Conference*, pp. 365-370, June 1997.

[4] 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.

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

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

[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] Q. Qiu, Q. Wu, C. 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.

[9] 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.

[10] 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.

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

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