Power Estimation of Behavioral Descriptions

Fabrizio Ferrandi * Franco Fummi * Enrico Macii ‡ Massimo Poncino ‡ Donatella Sciuto *

* Politecnico di Milano
Dip. di Elettronica e Informazione
Milano, ITALY 20133

‡ Politecnico di Torino
Dip. di Automatica e Informatica
Torino, ITALY 10129

Abstract

This paper presents a methodology for power estimation of designs described at the behavioral-level as the interconnection of functional modules. The input/output behavior of each module is implicitly stored using BDDs, and the power consumed by the network is estimated using a novel and accurate entropy-based approach. As a demonstration example, we have used the proposed power estimation technique to evaluate and compare the effects of some architectural transformations applied to a reference design specification on the power dissipation of the corresponding implementations.

1 Introduction

The increased degree of automation of industrial design frameworks has produced a substantial change in the way digital ICs are developed by the semiconductor companies. Today's systems are designed starting from specifications given at a very high level of abstraction (architectural or behavioral). This is because existing EDA tools are able to accept, as input, the description of a design expressed through an high-level HDL (e.g., VHDL, Verilog), and to automatically produce the corresponding low-level (gate or transistor) implementation, with very limited need of human intervention.

Since it is widely recognized that power consumption has become a critical issue in the development of digital systems — consider, for example, its impact on circuit complexity, speed, and manufacturing costs — designers from the semiconductor industry are demanding to the EDA vendors the development of computer programs and tools that allow to explicitly control the power budget during the various phases of the design process. This is because the power savings obtainable through automatic optimization have appeared to be less expensive than the ones achievable by resorting to technological choices (e.g., process and supply-voltage scaling).

The high-level section of a typical power-oriented design flow is depicted in Figure 1. Given an initial specification of the system behavior, several synthesis and optimization steps are required to generate a power-efficient netlist of logic gates. In order to make the search of the optimal solution as effective as possible, at each level of abstraction an "improvement loop" is used. In such a loop, a power estimator ranks the various design options, thus helping in selecting the one which is potentially more effective in terms of power. Obviously, collecting the feedback on the impact of the different choices on a level-by-level basis, instead of just at the very end of the flow (i.e., at the gate-level) allows a shorter development time. On the other hand, this paradigm requires the availability of *power estimation tools* which can provide accurate and reliable power figures at the various levels of abstraction.

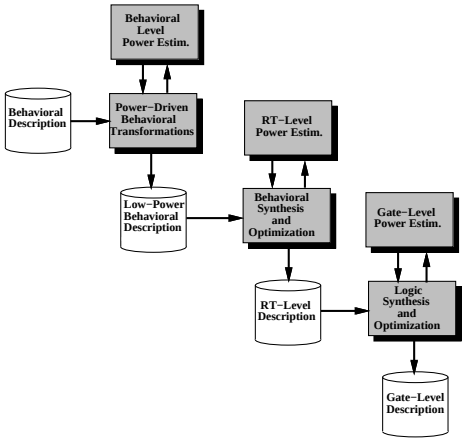

Figure 1: Low-Power Design Flow.

Most of the research on power estimation has initially focussed on gate and transistor levels; here, due to the available information on the structure and the macroscopic parameters of the devices (e.g., area, delay, capacitance), absolute power estimates can be determined; satisfactory methods, in terms of the accuracy in the estimation they can provide, have been obtained (see [1] for a detailed review).

More recently, some techniques for behavioral and architectural power estimation have been proposed (see [2] for a comprehensive survey). These techniques are usually based on the construction and the evaluation of abstract *power models*. This information is supposed to guide the designer in exploring the *relative* impact of his/her choices on the quality of the final design rather than to provide absolute power data. This is because, at the high levels of abstraction, the limited knowledge of the final structure of the design prevents the possibility of extracting meaningful power values.

In this work, we propose a novel approach to the problem of estimating the power dissipated by a digital design described at the behavioral-level as a network of interconnected functional modules. The I/O characteristics of the modules are implicitly stored using BDDs, and used as the input for a new and very efficient entropy-based power estimator. Notice that the term "behavioral" in this context has the meaning of "cycle accurate", that is, it indicates that only the I/O characteristics of the modules in the description are available.

We show how the proposed methodology can be used to estimate the power consumption of datapath-intensive designs, where the relative power figures are used for selecting design alternatives at the architectural-level, and we present some results to demonstrate the applicability of our approach to meaningful examples.

2 Power Estimation

2.1 Background and Previous Work

High-level power estimation approaches essentially belong to two categories: *Analytical* and *empirical* [2]. Analytical methods try to build a power model that can be expressed as a closed formula, in terms of a (possibly limited) number of macroscopic parameters. Parameters may include switching and statistical information, as well as technological quantities.
Empirical methods are based on the so-called *macro-modeling* approach. The power model still includes some parameters, generally of the same type as those used for the analytical models; however, they are now "modulated" by a set of coefficients. The actual model is evaluated by first selecting a proper model structure (e.g., linear), and eventually by measuring the actual power (e.g., through accurate simulation) for different assignments of the parameters. These assignments yield a set of points in a bi-dimensional space (actual power vs. power from the model) that can be inter- or extra-polated to get the final equation of the model.
Empirical models are usually preferable; in fact, thanks to their derivation from the actual power measures, they are more accurate. However, they tend to be complex in terms of the number of required parameters; thus, they may take quite long characterization times. Analytical models, on the other hand, are normally less accurate. However, they are much simpler than the empirical ones, and they can be used for black-box estimation, that is, in the cases where little or no information about the internal structure of the modules is available, or when (as for libraries of *soft-macros*) synthesis can not be used before the characterization.
The methodology we propose features a "mixed" solution: On the one hand, it sticks to an entropy-based analytical model for the switching activity component of the power; on the other hand, it resorts to an empirical model, still entropy-based, for capacitance estimation. The advantages of our approach are the following:

- Low complexity: The proposed models always have at most four parameters;
- Abstraction: The parameters only concern the I/O behavior of the modules, and they do not refer to the internal structure;
- Accuracy: Using empirical capacitance models sensibly improves the quality of the estimates with respect to previous entropy-based approaches.

2.2 Entropy and Power Dissipation

2.2.1 Power Dissipation Model

At the gate-level, the average power consumption of a static CMOS gate is given by:

$$P^g_{avg} = \frac{1}{2} \cdot C_g \cdot \frac{V^2_{dd}}{T_C} \cdot E_g = \frac{1}{2} \cdot C_g \cdot V^2_{dd} \cdot D_g, \qquad (1)$$

where C_g is the capacitance of the output load, V_{dd} is the supply voltage, T_C is the global clock period, and E_g is the average number of gate output transitions per clock cycle. According to the notation of [3], factors E_g and $\frac{1}{T_C}$ can be lumped together to get the *transition density* D_g. The total average power is then obtained by summing the contributions of Equation 1 over all the N gates in the circuit:

$$P_{avg} = \frac{1}{2} \cdot V^2_{dd} \cdot \sum_{i=1}^{N} C_i \cdot D_i. \qquad (2)$$

At the behavioral-level, some approximations must be introduced to account for the limited knowledge of the circuit structure. At this level, the interest goes more into an average estimate per module, rather than into a gate-by-gate information. Therefore, Equation 1 can be rewritten as [3]:

$$P_{avg} \propto \mathcal{D} \cdot C_{tot}, \qquad (3)$$

where $\mathcal{D} = \frac{1}{N} \sum_{i=1}^{N} D_i$ and $C_{tot} = \sum_{i=1}^{N} C_i$ represent the average transition density and the total capacitance of the module, respectively.
The key point is to find some simplifications that allow to express $\mathcal{D}$ and C_{tot} as functions of only the I/O behavior of each module, independently of the internal structure of the implementation.

2.2.2 Entropy of a Logic Function

Given a n-input, m-output Boolean function, $F(f_1, \ldots, f_m)$, where $f_k = f_k(X) = f_k(x_1, \ldots, x_n)$, $k = 1, \ldots m$, the *input entropy* of F, denoted as $H_I(F)$, is given by:

$$H_I(F) = \sum_{i=1}^{2^n} p_i \log_2 \frac{1}{p_i}, \qquad (4)$$

where p_i indicates the probability of the input vector X to take on the value X_i.
The *output entropy* of function F, denoted as $H_O(F)$, is given by:

$$H_O(F) = \sum_{i=1}^{2^m} q_i \log_2 \frac{1}{q_i}, \qquad (5)$$

where q_i indicates the probability of the output vector to assume the value O_i. Obviously, $q_i = n_{O_i}/2^n$, where n_{O_i} denotes the number of occurrences of the vector O_i in the truth table of F. It can be noticed that $H_O(F) < H_I(F)$; this is true in general, that is, given any Boolean function, F, we have that $H_O(F) \leq H_I(F)$ [4].

2.2.3 Computing the Average Transition Density

In [3, 5], it has been shown that the average density $\mathcal{D}$ of a module can be approximated as:

$$\mathcal{D} = k \cdot \mathcal{H} \qquad (6)$$

where k represents a proportionality factor, and $\mathcal{H}$ represents the average entropy for the module.
$\mathcal{H}$ can be computed by abstracting information obtained from a gate-level implementation. By introducing different approximations, the two methods result in different values for $\mathcal{H}$. The simplest one, given in [3], determines $\mathcal{H}$ for a module with n inputs and m outputs as follows:

$$\mathcal{H} = \frac{2}{3(n+m)}(H_I + H_O) \qquad (7)$$

where H_I and H_O are the input and output entropies of the module, and they are calculated accounting for the input/output statistics.
Although the formula above allows to correlate the statistical behavior of a module to the corresponding I/O behavior, computing H_O directly from the definition of Equation 5 is infeasible for reasonably large circuits; similarly, the computation of H_I gets difficult for a module whose input statistics depend on the output statistics of some other modules.

In [3, 5], this problem has been solved by simply assuming the statistical independence of the module outputs; the output entropy of a multi-output module is thus computed as the sum of the entropies associated to each output.
A symbolic method for computing the output entropy of a multi-output circuit without resorting to the definition of Equation 5 has been proposed in [6]. The technique can determine exact entropy values for functions of reasonable size; however, when the computational effort gets too high, it is possible to trade off accuracy for time by selectively grouping outputs in clusters, by computing and summing the entropy contributions of each cluster, and thus obtaining an approximate entropy result [7].

2.2.4 Computing the Total Capacitance

Accurate capacitance values are obviously available only after mapping the circuit onto a specified gate-library. At the technology-independent level, under the simplifying assumption that each transistor in the final implementation will contribute the same load, a reasonable approximation can be obtained by relating the capacitance of a module to the number of literals in the gate-level realization.
To express this relation in the entropy domain, we can use an empirical result reported in [8]. We have that, for a n-input Boolean function ($n \geq 10$):

$$C_{tot} \propto \frac{2^n}{n} \cdot H_O \tag{8}$$

since the number of literals of the function is proportional to $\frac{2^n}{n} H_O$.

2.3 Power Estimation Methodology

The `ENTEST` program carries out the power estimation of an interconnection of datapath modules described with the `arc` format. This format provides a very simple syntax for specifying an arbitrary interconnection of both combinational and sequential modules. In our context, `arc` descriptions are automatically generated starting from an algorithmic description of the design given in VHDL [9].
Besides the specification of the system topology, an `arc` description also contains, for each module, pointers to information of two types:

- Structural, when available, provided through a structural description language; currently, only the `blif` language is supported;
- Functional, for behavioral descriptions, stored as BDDs. More precisely, for each module, one BDD is stored for each output.

`ENTEST` also accepts an input probability file, to be used for entropy computation. This file contains the statistics of the external inputs of the overall system, given as probabilities for the individual signals. If no file is provided, the external inputs are assumed to be statistically independent (i.e., to have a random distribution). Notice that, in a more general context, these signal probabilities can be extracted from typical input streams determined through system-level simulation or provided together with the simulatable system description. Finally, `ENTEST` also reads a target gate library that contains the information gathered during the characterization phase (see Section 2.3.3), namely, the coefficients of the power model for the given library. Figure 2 shows a block diagram describing the input/output interface of `ENTEST`.

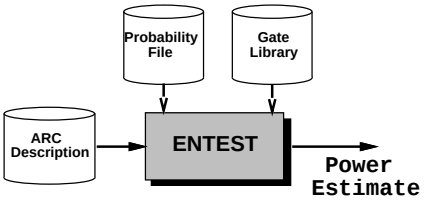

Figure 2: Behavioral Power Estimator.

Starting from the `arc` description and the probability file, the program proceeds as follows. First, the network of interconnected modules is traversed breadth-first, starting from the external inputs. This is equivalent to levelizing the network. Then, the implicit algorithm of [6] is applied, on a module-by-module basis and following the breadth-first order, to compute exact input and output entropies for each module. The choice of considering modules in breadth-first order is mandatory to guarantee that the input statistics for each module are known before computing the corresponding entropy value.
When the exact computation for some modules fails, due to size reasons, approximate entropy values are determined for such modules; this is done as in [7], that is, by partitioning the set of outputs of each module in clusters of manageable size. The partitioning algorithm tries to group together outputs having maximal pairwise *correlation*, i.e., maximal similarity in the Boolean space [10].
A further problem to be considered is that the input statistics of modules inside the network can not be assumed to be known a priori as in the case of external inputs. Therefore, since the topology of the network of modules introduces correlation between signals, the input statistics of each module change, and must be derived from the output statistics of the upstream modules. To do this, a specialized version of the procedure of [6] must be used.
The above approach for average entropy computation sensibly improves over previous methods [3, 5] in two main directions: First, an exact entropy value (or a more accurate approximation) is calculated for each module; second, entropy values are computed according to the actual input statistics of the modules, rather than considering each module in isolation, as in previous contributions. The latter assumption is a significant source of errors, since the formula which gives the average entropy $\mathcal{H}$ (Equation 7) also depends on the input entropy.

2.3.1 Switching Activity Estimation

Estimation of the switching activity simply consists of plugging the exact entropies, or the entropy upper bounds, computed as discussed in the previous section, into a model similar to that of Equation 7.
From Equation 3, the estimate of the transition density is assumed to be proportional to both capacitance value and average entropy. However, since the average entropy $\mathcal{H}$, as in Equation 7, is a number between 0 and 1, the impact of entropy (i.e., switching activity) on the product $\mathcal{H} \cdot \mathcal{A}$ can be thought of as a factor that "modulates" the capacitance values. Since the latter is a number (and not a probability), we have chosen a simple analytical model as that of Equation 7 instead of developing a new model for average entropy (or switching activity); conversely, we have elaborated a more accurate model for capacitance estimation, since this has proved to be the weakest point of existing entropy-based power estimation approaches.

2.3.2 Capacitance Estimation

As already mentioned, the problem of estimating the capacitance can be reduced to that of predicting the area of the implementation by establishing a relation between the number of literals and the input/output behavior, as shown in Equation 8. In [3], it has been observed that this model is quite inadequate for typical VLSI functions. For example, a 32-input module should require approximately 2^{27} literals to be implemented, which is clearly an unrealistic estimate. Improved solutions to overcome the limitations of this model [11] did not provide a substantial breakthrough in the refinement of the estimate.

Unlike similar methods, the `ENTEST` program implements an effective area estimation procedure that exploits the additional information contained in the BDD representation. Although it is common belief that there is no clear correlation between the size of a gate-level implementation and the size (i.e., the number of nodes) of the corresponding BDD representation, we have used the latter information as a link between the size of the implementation and its representation. The rationale behind this idea comes from the following observation: Since we consider multi-output circuits, there is a considerable amount of sharing of the internal BDD nodes, which somehow resembles the sharing of logic in the multi-level logic implementation. Furthermore, since we can think of each BDD node as a 2-to-1 multiplexor, with the corresponding variable as control input, we can roughly claim that each multiplexor corresponds to a fixed number of gates in the library. Based on this intuition, the model we propose for estimating the total capacitance is:

$$C_{tot} = a \cdot N_{nodes} \cdot \frac{H_O}{m} + b \tag{9}$$

where the exponential factor in Equation 8 has been replaced by the more realistic value of the number N_{nodes} of nodes of the BDD of the multiple-output function of the module. Parameter m is the number of module outputs, and it is used to normalize the contribution of H_O. The reason why Equation 8 overestimates the size of the implementation can be found in the factor $\frac{2^n}{n}$, exponential in the number of inputs, that represents the worst-case size of the circuit, as well as the worst-case number of BDD nodes.

The coefficients of the linear model of Equation 9 have been determined empirically by resorting to *macro-modeling*; we have selected a large set of benchmark circuits, optimized them for minimum-area, and mapped them onto a library of gates. Then, we have plotted the number of gates of the mapped network and the corresponding number of BDD nodes in a two-dimensional space, and we have used least mean square (LMS) regression to derive the values of the coefficients a and b.

As shown in the diagram of Figure 3, there is a clear linear relation (in logarithmic scale) between the two values. There are a few points in the diagram for which the LMS line yields significant errors. The analysis of these cases has shown that the singularities are due to very particular functions which can be indeed classified as exceptions. The points below the interpolation line represent the single-output functions of the benchmark set; for the reason described above, the limited sharing of nodes with the BDDs of other outputs can not model properly the sharing of logic as imposed by the optimization step. The points above the line represent functions for which the BDD representation is way smaller than the logic implementation. Since the circuits we expect to deal with are datapath modules, we believe that they will fall in the class of "regular" circuits; then, they will be accurately modeled by the estimator.

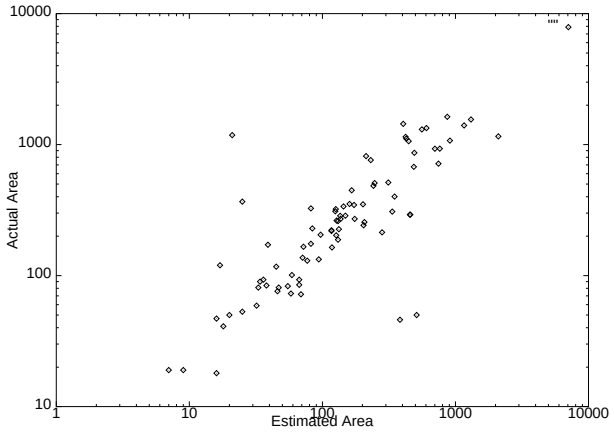

Figure 3: Estimated vs. Actual Area.

One potential drawback of the model above is the sensitivity of the BDD size to the input variable ordering. BDDs with different orderings, but representing the same function, may have totally different node counts. Then, since the modules we are considering usually have more than one output, a good ordering for one BDD may be bad for another one, and vice versa. But what a dynamic reordering procedure looks for is an ordering for which the size of the overall shared BDD is minimized; thus, the impact of the variable ordering on the model is partially smoothed out by the fact that the modules have multiple outputs. Nevertheless, to avoid the occurrence of undesired situations, the parameter N_{nodes} in Equation 9 is actually a weighted average between the size of the BDD as given in the original module description and the minimum size as obtained after sifting-based reordering.

Finally, notice that the values on the y-axis in Figure 3 depend on the choice of the library of gates used for the characterization process, and such a choice may affect the quality of the estimates. As a matter of fact, each available library has a different pair of the (a, b) coefficients, which are determined as described above, once and for all, when the selected library is loaded for the first time. The plot in Figure 3 is referred to a simple 2-input NAND-NOR library.

2.3.3 Power Estimation

Once both entropy and capacitance estimations are available, their values are plugged into the model of Equation 3. In order to correlate values of estimated power to actual power, a proper proportionality factor must be computed. The purpose is to derive an equation which works as an estimator, expressing power consumed by a module as a function of the input and output entropies, or indirectly, of the $\mathcal{H} \cdot \mathcal{C}$ product.

Similarly to the case of capacitance, a LMS regression is applied to the diagram of the estimated power versus the actual power. Actual power here is referred to estimates determined using Synopsys Design Power. Figure 4 shows the diagram obtained using the same set of benchmarks as the one used for capacitance estimation.

The LMS regression line (in log scale) can be directly used as an estimator for the generic modules instantiated in a given architecture.

As an example, for the reference 2-input NAND-NOR library used in the experiments, the LMS in logarithmic scale yields the following estimator equation: $P_{est} = 217.7 \cdot x^{0.737}$, where x denotes the independent variable, that is, the $\mathcal{H} \cdot \mathcal{C}$ product.

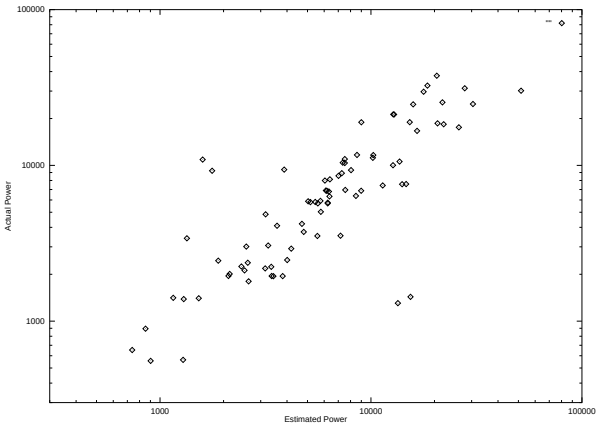

Figure 4: Estimated vs. Actual Power.

3 Experimental Results

In order to evaluate the effectiveness of the proposed power estimation methodology, we have experimented with some designs described at the behavioral-level as networks of interacting modules. We report the results we have obtained on two of them. The first one is the differential equation solver taken from the HLSynth'92 benchmarks suite [12]; the second example is an industrial microcontroller described as a re-configurable macro. For each design, different architectures have been considered starting from the reference specification, and the corresponding power estimates have been used to rank the various solutions.

Concerning the differential equation solver, the first variant to the reference architecture has been obtained by modifying the expression of a signal in the description through rewriting. The new datapath, denoted as `diffeq-a`, uses two less modules than the reference specification, and it is thus expected to consume less power. The second alternative, indicated as `diffeq-b`, has been created by replacing a multiplication with a shift-and-add operation. This type of transformation usually helps in reducing power consumption [13]. The last solution, indicated as `diffeq-c`, is generated by increasing the amount of sharing across the modules, which reduces the amount of hardware units used. In particular, a sub-expression common to two signals has been factored. Again, this modification should result in a decrease in power, due to the use of fewer computational units.

Regarding the industrial microcontroller, two architectures have been derived from the reference one by assigning different values to some of the configuration parameters. In particular, the first solution, denoted as `micro-a`, has been obtained by increasing from 16 to 32 the bit width of the status register. The second architecture, called `micro-b`, has been obtained by decreasing from 8 to 4 the number of interrupt request lines that are managed by the interrupt controller. Clearly, while the first architecture is expected to be more power consuming than the reference one, the second alternative should be more power efficient.

The relative power results with respect to the reference solutions are reported on the left-hand side of Table 1 (column `ENTEST`). The data confirm the expected trends of relative power.

In order to validate the power figures determined at the high level, each architecture has been synthesized onto a library of gates, and the actual power dissipated by each gate-level implementation has been estimated using Design Power. The so obtained results, again expressed in relative terms with respect to the reference circuit descriptions, are reported on the right-hand side of Table 1 (column `Design Power`).

The data in the table show, with surprising accuracy, that all the trends of relative power have been conservatively preserved, as it can be observed by comparing the values in the two columns for a given design.

Design	*Architecture*	*Relative Power [%]*	
		`ENTEST`	`Design Power`
`diffeq`	`diffeq-ref`	100.0	100.0
	`diffeq-a`	81.0	75.8
	`diffeq-b`	92.3	93.4
	`diffeq-c`	93.1	95.4
`micro`	`micro-ref`	100.0	100.0
	`micro-a`	142.0	150.8
	`micro-b`	78.1	76.4

Table 1: Relative Power Results.

4 Conclusions

We have presented a method for estimating the power dissipated by a behavioral description given as the interconnection of modules. The technique relies on an improved entropy-based approach for switching activity and capacitance evaluation.

The proposed method is particularly suited for power estimation of datapath-intensive systems, where the estimates are used for selecting design alternatives at the behavioral-level. Experiments carried out on some realistic designs have shown the effectiveness of the approach as a relative indicator; our estimates ranked the descriptions resulting from the various architectural transformations in the same order as those obtained with post-synthesis estimates.

Acknowledgments

We wish to thank CSELT and SGS-Thomson for providing us with the `micro` benchmark.

References

[1] F. N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Trans. on VLSI Systems*, Vol. 2, No. 4, pp. 446-455, December 1994.

[2] P. Landman, "High-Level Power Estimation," *ISLPED-96*, pp. 29-35, Aug. 1996.

[3] M. Nemani, F. Najm, "Towards a High-Level Power Estimation Capability," *IEEE Trans. on CAD*, Vol. 15, No. 6, pp. 588-598, Jun. 1996.

[4] T. M. Cover, J. A. Thomas, *Elements of Information Science*, New York, NY, John Wiley and Sons, 1991.

[5] D. Marculescu, R. Marculescu, M. Pedram, "Information Theoretic Measures For Power Analysis," *IEEE Trans. on CAD*, Vol. 15, No. 6, pp. 599-609, Jun. 1996.

[6] E. Macii, M. Poncino, "Exact Computation of the Entropy of a Logic Circuit," *GLS-VLSI-96*, pp. 123-128, Mar. 1996.

[7] A. Lioy, E. Macii, M. Poncino, M. Rossello, "Accurate Entropy Calculation for Large Logic Circuits Based on Output Clustering," *GLS-VLSI-97*, pp. 70-75, Mar. 1997.

[8] K. T. Cheng, V. D. Agrawal, "An Entropy Measure for the Complexity of Multi-Output Boolean Functions," *DAC-27*, pp. 302-305, Jun. 1990.

[9] G. Buonanno, F. Ferrandi, L. Ferrandi, F. Fummi, D. Sciuto, "How an Evolving Fault Model Improves the Behavioral Test Generation," *GLS-VLSI-97*, pp. 124-129, Mar. 1997.

[10] H. Cho, G. D. Hachtel, E. Macii, M. Poncino, F. Somenzi, "Automatic State Space Decomposition for Approximate FSM Traversal Based on Circuit Structural Analysis," *IEEE Trans. on CAD*, Vol. 15, No. 12, pp. 1451-1464, Dec. 1996.

[11] M. Nemani, F. Najm, "High-Level Power Estimation and the Area Complexity of Boolean Functions," *ISLPED-96*, pp. 329-334, Aug. 1996.

[12] *High-Level Synthesis Benchmarks*, CAD Benchmarking Laboratory (CBL), North Carolina State University, 1992.

[13] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, R. W. Brodersen, "Optimizing Power Using Transformations," *IEEE Trans. on CAD*, Von. 14, No. 1, pp 12-31, Jan. 1995.