Regression-Based Macromodeling for Delay Estimation of Behavioral Components

A. Macii E. Macii G. Odasso M. Poncino R. Scarsi

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

Abstract

This paper presents a methodology for delay estimation of hardware components described at the behavioral-level. The basis of the proposed technique is a well-known theoretical result that relates the entropy of a logic function to the delay of a multi-level implementation of the same function. We propose an improved model for delay estimation, and we prove its validity by means of experiments performed on a set of standard benchmarks.

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 high level of abstraction (e.g., behavioral). EDA tools are then 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.

The high-level section of a typical 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 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, proper estimators rank the various design options, thus helping in selecting the one which is potentially more effective in terms of the chosen cost function (e.g., area, delay, 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 *estimation tools* that can provide accurate and reliable quality figures at the various levels of abstraction.

In this work, we address the problem of estimating the delay of a digital component described at the behavioral level. This is a central problem in modern VLSI circuit development; in fact, performance constraints have the highest priority among the various metrics that are normally considered during the design process. It is key to observe that here the term "behavioral" has the meaning of "cycle accurate", that is, it indicates that only the I/O characteristics of the component under analysis are available. Therefore, details concerning the actual implementation of such component are not required.

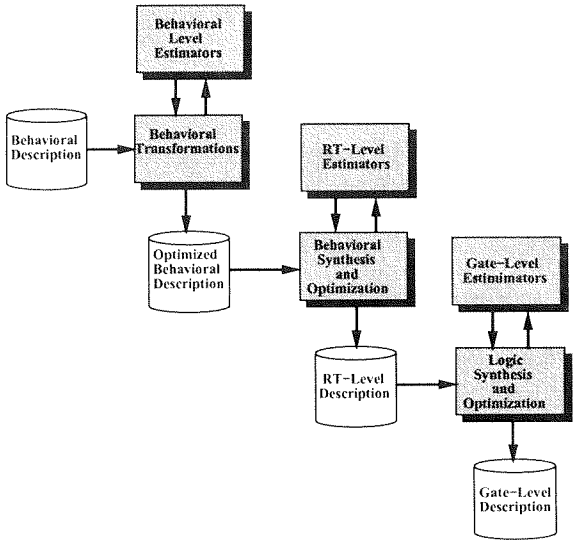

Figure 1: High-Level IC Design Flow.

We propose a delay estimation technique based on *macro-modeling* that is similar to those usually adopted for high-level power estimation (see [1] for a survey on this subject). One of the most difficult tasks in this class of estimation approaches is the selection of the parameters to be included in the model. We contribute properly motivated guidelines on how the selection should be carried out when the target is delay modeling. Also, we show how the proposed solution can be successfully used to estimate the delay of complex components by applying it to the complete set of the Iscas'85 combinational benchmarks [2].

2 Previous Work

Several works on high-level delay modeling and estimation do exist in the literature. It is worth mentioning, among others (e.g., [3, 4, 5]), the approach proposed by Nemani

0-7695-0104-4/99 $10.00

and Najm in [6]. The key concept on which this method is based is that of *delay complexity* of a Boolean function: Given a single-output Boolean function, f, and called C^{on} the ON-set of f, the delay complexity of C^{on} is defined as:

$$\mathcal{M}_1(f) = \log_2(\max_{c_i \in C^{on}} |c_i|) + \log_2(|C^{on}|)$$

where $c_i \in C^{on}$ is a cube of C^{on}, $|c_i|$ is the number of literals in cube c_i, and $|C^{on}|$ is the number of cubes in C^{on}. Similarly, the delay complexity of C^{off} (i.e., the OFF-set of f) is defined as:

$$\mathcal{M}_0(f) = \log_2(\max_{c_i \in C^{off}} |c_i|) + \log_2(|C^{off}|)$$

Then, the *delay measure* of f is defined as:

$$\mathcal{M}(f) = \min(\mathcal{M}_1(f), \mathcal{M}_0(f))$$

A model of the minimum delay, $\mathcal{D}_{min}$, for a network N implementing function f is determined by empirical observation of the delay of randomly-generated single-output circuits having the same support as f:

$$\mathcal{D}_{min}(N) = k_1 \mathcal{M}(f) + k_2$$

where k_1 and k_2 are the coefficients of the model determined through least squares fit of the experimental measurements.
Similarly, the model for the maximum delay, $\mathcal{D}_{MAX}$, for a network N implementing function f is given by:

$$\mathcal{D}_{MAX}(N) = k_1 \beta^{\mathcal{M}(f)}$$

where $\beta \approx 1.3$ for the specific technology library considered for the experiments.
A simple extension of the model to the case of multiple-output Boolean functions, based on the idea of output multiplexing [7, 8] is also provided.
Results reported on the application of this method to several benchmarks circuits exhibit significant inaccuracies in delay estimates for circuits with lot of shared logic. This is because the load driven by multiple-fanout nodes is not properly taken into account by the model.

3 Background

3.1 Entropy of a Logic Function

Given a n-input, m-output Boolean function, $F(f_1, .., f_m)$, 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:

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

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:

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

where q_i indicates the probability of function F to assume the value O_i (O_i is a vector m Boolean values). Obviously, $q_i = n_{O_i}/2^n$, where n_{O_i} denotes the number of occurrences of vector O_i in the (m-bit wide) output column of the truth table of F.
A symbolic method for computing the output entropy of a multi-output function without resorting to the definition of Equation 2 has been proposed in [11]. The technique can determine exact entropy values for functions of reasonable size. For larger functions, approximate solutions can be used [12].

3.2 Macromodeling

A typical way to estimate some quantity (e.g., area, delay, power) from a high-level circuit description is to build an abstract model bottom-up from either a low-effort implementation of the circuit itself or from existing designs. These empirical models are usually called *macromodels*.
Rather than a conventional formula, a macromodel can be regarded as a fixed template in which the quantity Q under analysis is related to a set of parameters through a set of coefficients, that is:

$$Q = \mathcal{F}(parameters, coefficients)$$

The first step in the macromodeling paradigm is the design of the model. This amounts to choosing: *i*) The shape of the model (e.g., linear vs. non-linear). *ii*) The parameters to be considered. This step clearly depends on the quantity being modeled, but also on the degree of complexity allowed.
The second step is the *characterization* step, in which the model is tuned against some *training set*. Each element of this set identifies a point in the multi-dimensional parameter space (the independent variables), and is associated to a value of the quantity being modeled (the dependent variable), obtained by actually measuring it.
In the final step, techniques borrowed from statistics are used to extract the actual equation describing the model. More precisely, this is obtained by intra- or extra-polation of the points corresponding to the training set. The most typical solution resorts to linear regression analysis [1], which is usually preferred to more sophisticated models thanks to its well-established mathematical foundations.
Due to its statistical nature, one key issue in macromodeling is the representativity of the training set. Since the model is built from a set of samples, such set should span the parameter space as much as possible to prevent the model from being too sensitive to the training conditions.

4 Macromodeling for Delay Estimation

We assume that the functional specification of the circuit for which delay estimation must be performed is given through an array of binary decision diagrams (BDDs) [13], each of which represents the logic function implemented by each output.
According to [14], a reasonable estimate of the maximum delay for a network N implementing a n-input, single-

output Boolean function, f, is given by:

$$\mathcal{D}(N) = k_D 2^{\frac{n}{2}} H_O^{\frac{1}{2}}(f) \quad (3)$$

where k_D is a proportionality factor, and $H_O(f)$ is the output entropy of function f.

The simple model above, inaccurate by itself, cannot be easily generalized to a network N implementing a multi-output function $F = (f_1, \ldots, f_m)$. Unlike the case of area estimation, replacing term $H_O(f)$ in Equation 3 with the normalized output entropy with $\frac{H_O(F)}{m}$ is far from providing a satisfactory model.

The main reason for this is that delay, unlike area, is not strictly proportional to circuit complexity. A large number of outputs not necessarily implies a longer delay. In addition, considering all the outputs is not very meaningful in the context of delay prediction, because only the most relevant (in terms of delay) outputs should be taken into account. Finally, there are other factors that contribute very heavily to the delay of a circuit output: Besides the complexity of the function expressed in terms of the quantity of information it carries (entropy is a good way of modeling this contribution), there is the implementation complexity (e.g., number of equivalent gates), the size of the support (usually, the larger the support size, the deeper the cone of logic feeding the output), and the density of the function (a factor that somehow accounts for the distance of the function from an equivalent, minimum-size two-level implementation).

In view of this discussion, the model we propose for the delay of a network N implementing a n-input, m-output function $F = (f_1, \ldots, f_m)$ is the following:

$$\mathcal{D}(N) = k_0 + k_1 H_O^{\frac{1}{2}}(F_S) + k_2 \log_2\left(\frac{\sum_{i=1}^{y} Size(f_i)}{y}\right) + k_3 \frac{\sum_{i=1}^{y} |Support(f_i)|}{y} + k_4 \frac{\sum_{i=1}^{y} Density(f_i)}{y} \quad (4)$$

where:

- $F_S = (f_1, \ldots, f_y)$ is a n-input, y-output function obtained from F by removing some selected outputs; the details on how F_S is chosen are given later in this section.
- $H_O^{\frac{1}{2}}(F_S)$ is the square root of the output entropy of function F_S.
- $Size(f_i)$ is the number of BDD nodes of single-output function f_i.
- $|Support(f_i)|$ is the number of inputs f_i actually depends on.
- $Density(f_i)$ is the density of single-output function f_i, whose definition is the following:

  $$Density(f_i) = Abs(Prob(f_i) - 0.5)$$

  where $Prob(f_i)$ is the fraction of minterms in the ON-set of f_i.

The choice of function F_S is key for the model accuracy. As discussed earlier, only the outputs with potentially longest delays should be considered during the estimation. Since the major impact on the actual delay of a single-output function is caused by both its support and its size, we identify the elements of set $(f_1, \ldots, f_y)$ as those functions having the largest values of the product

$$SS = \log_2(Size) \cdot |Support|;$$

more precisely, we compute the maximum SS value, and pick all the f_i's whose SS is within a given percentage of the maximum value (30% in our case).

The coefficients of the macromodel of Equation 4 have been determined empirically as follows: As training set we have selected a large set of circuits taken from the Mcnc'91 suite [9], optimized them for minimum-delay, and mapped them onto a library of gates. Then, we have used a multi-dimensional regression to derive the values of the coefficients $k_0, \ldots, k_4$.

For technology mapping we have used a simplified version of the `lib2.genlib` library distributed along with the `SIS` synthesis system [15], which does not include `aoi` and `oai` cells.

Figure 2 shows the plot of the estimated versus actual delays for the benchmarks used for characterizing the model.

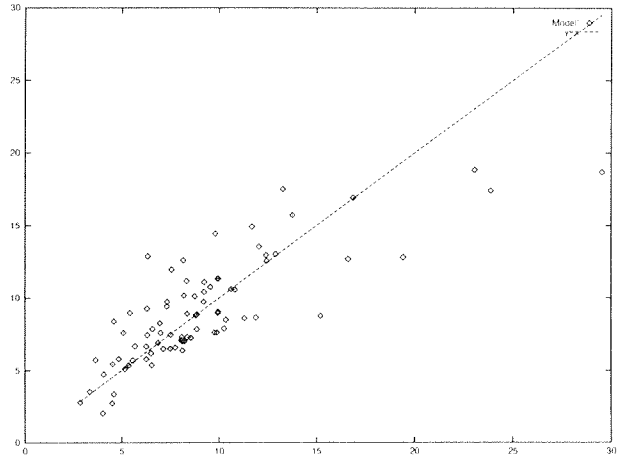

Figure 2: Actual (x-axis) vs. Estimated (y-axis) Delay.

One potential drawback of the model of Equation 4 is the sensitivity of the BDD size to the input variable ordering [16]. BDDs that represent the same function, but with different orderings, may have totally different node counts. Since we are considering several outputs at the same time, 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 we are considering multiple outputs. Nevertheless, to avoid the occurrence of undesired situations, parameter $Size(f_i)$ in Equation 4 is actually a weighted average between the size of the BDD as given in the original component description and the minimum size as obtained after sifting-based reordering [17].

5 Experimental Results

We have applied the delay estimation technique described in Section 4 to all the Iscas'85 benchmarks [2]. BDD representations have been obtained from the Boolean network specifications of the circuits (i.e., technology-independent gate-level netlists) through the functions provided by the CUDD package [18]. Delays have then been estimated using the macromodel of Equation 4, and the results of the calculation are reported in column *Model* of Table 1.
In order to validate the delay figures determined at the behavioral-level using the proposed macromodel, each circuit has been mapped onto the simplified `lib2.genlib` technology library described in Section 4, and the actual delay of the gate-level implementation has been estimated using SIS [15]. The so obtained results are reported in the column *SIS* of Table 1.

Circ.	*Description*	*I*	*O*	*Delay*		*Err*
				Model	*SIS*	[%]
c432	Interr. Contr.	36	6	16.9	27.1	37.4
c499	Comparator	41	32	28.1	19.4	44.6
c880	ALU	60	26	31.6	18.8	68.2
c1355	Comparator	41	32	27.7	21.0	32.0
c1908	Error Corr.	33	25	24.3	25.8	5.9
c2670	ALU+Contr.	233	140	18.3	25.5	28.3
c3540	ALU	50	22	32.0	36.1	11.3
c5315	ALU	178	123	15.0	30.2	50.5
c7552	ALU	207	108	15.1	26.2	42.4
Avg.						35.6

Table 1: Delay Estimation Results.

Only one benchmark is missing from the table: c6288. This circuit is a 32-bit multiplier, for which it is well known that the computation of the BDDs for all the outputs is infeasible [19].
Results, although quite preliminary, are encouraging. In fact, it should be considered that the proposed model considers only purely behavioral parameters, and therefore it can be applied very early in the design flow for the purpose of design exploration. In this situation, absolute estimation errors of several orders of magnitude are usually acceptable, since predictions made at this stage are only needed for relative comparisons of different design options.

6 Conclusions

We have contributed a technique for estimating the delay of components described at the behavioral-level. The proposed approach relies on an improved macromodel whose main feature is the exploitation of parameters that relate various types of complexity of a Boolean function to the delay of its gate-level implementation. Experiments carried out on some standard benchmarks have shown the effectiveness of the presented solution.

References

[1] E. Macii, M. Pedram, F. Somenzi, "High-Level Power Modeling, Estimation, and Optimization," *IEEE Trans. on CAD*, Vol. 17, No. 11, November 1998.

[2] F. Brglez, H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," *ISCAS-85*, pp. 785-794, Kyoto, Japan, June 1985.

[3] C. Ramachandran, F. Kurdahi, "Combined Topological and Functionality-Based Delay Estimation Using a Layout-Driven Approach for High-Level Applications," *IEEE Trans. on CAD*, Vol. 13. No. 2, pp. 1450-1460, December 1994.

[4] H. Yalcin, J. Hayes, K. Sakallah, "An Approximate Timing Analysis Method for Datapath Circuits," *ICCAD-96*, pp. 114-118, San Jose, CA, November 1996.

[5] A. Srinivansan, G. D. Huber, D. P. LaPotin, "Accurate Area and Delay Estimation form RTL Descriptions," *IEEE Trans. on VLSI Systems*, Vol. 6, No. 1, pp. 168-172, March 1998.

[6] M. Nemani, F. N. Najm, "Delay Estimation of VLSI Circuits from a High-Level View," *DAC-35*, pp. 591-594, San Francisco, CA, June 1998.

[7] M. Nemani, F. Najm, "High-Level Area Prediction for Power Estimation," *CICC-97*, pp. 483-486, Santa Clara, CA, May 1997.

[8] M. Nemani, F. Najm, "High-Level Area and Power Estimation for VLSI Circuits," *ICCAD-97*, pp. 114-119, San Jose, CA, November 1997.

[9] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide Version 3.0*, Technical Report, MCNC: Microelectronics Center of North Carolina, Research Triangle Park, NC, January 1991.

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

[11] E. Macii, M. Poncino, "Exact Computation of the Entropy of a Logic Circuit", *GLS-VLSI-96*, pp. 162-167, Ames, Iowa, March 1996.

[12] 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, Urbana-Champaign, Illinois, March 1997.

[13] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. on Computers*, Vol. 35, No. 8, pp. 677-691, August 1986.

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

[15] E. M. Sentovich, K. J. Singh, C. W. Moon, H. Savoj, R. K. Brayton, A. Sangiovanni-Vincentelli, "Sequential Circuits Design Using Synthesis and Optimization," *ICCD-92*, pp. 328-333, Cambridge, MA, October 1992.

[16] G. D. Hachtel, F. Somenzi, *Algorithms for Logic Synthesis and Verification*, Kluwer Academic Publishers, 1996.

[17] R. Rudell, "Dynamic Variable Ordering for Ordered Binary Decision Diagrams", *ICCAD-93*, pp. 47-47, Santa Clara, CA, November 1993.

[18] F. Somenzi, *CUDD: University of Colorado Decision Diagram Package*, Release 2.3.0, Technical Report, Dept. of ECE, University of Colorado, Boulder, CO, September 1998.

[19] R. E. Bryant, "On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication," *IEEE Trans. on Computers*, Vol. 40, No. 2, pp. 205-213, February 1991.