High-Level Area Prediction for Power Estimation†

Mahadevamurty Nemani and Farid N. Najm

ECE Dept. and Coordinated Science Lab.
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Abstract

High-level power estimation, when given *only* a high level design specification such as a functional or RTL description, requires high-level estimation of the circuit average activity and total capacitance. Considering that total capacitance is related to circuit area, this paper addresses the problem of computing the area complexity of single-output Boolean functions given only their functional description, where area complexity is measured in terms of the number of gates required for an *optimal* implementation of the function. We propose an area model that makes use of a new complexity measure. The model is empirical, and is based on an observed relationship between the proposed complexity measure, which is easily measurable using Monte-Carlo simulation, and its *optimal* implementation (gate-count). This model has been implemented, and empirical results demonstrating its feasibility and utility are presented.

1. Introduction

Rapid increase in the design complexity and reduction in design time have resulted in a need for CAD tools that can help make important design decisions *early* in the design process. To do so, these tools must operate with a design description at a high level of abstraction. One design criterion that has received increased attention lately is power dissipation. As a result, there is a need for *high level power estimation* and optimization. Specifically, it would be highly beneficial to have a power estimation capability, given only a *functional* view of the design, such as when a circuit is described only with Boolean equations. Of course, a given Boolean function can be implemented in many ways, with varying power dissipation levels. We are interested in predicting the *nominal* power dissipation that a minimal area implementation of the function would have.

For a combinational circuit, since the only available information is its Boolean function, we consider that its power dissipation will be modeled as follows:

$$P_{avg} = \mathcal{D}_{avg}\mathcal{A}\mathcal{C}_{avg} \quad (1)$$

where $\mathcal{D}_{avg}$ is an estimate of the average node switching activity that a gate-level implementation of this function would have, $\mathcal{A}$ is an estimate of the gate count (assuming some target gate library), and $\mathcal{C}_{avg}$ is an estimate of the average gate capacitance (including drain capacitance and interconnect loading capacitance).

† This work was supported in part by Intel Corporation, and by the National Science Foundation.

The estimation of $\mathcal{D}_{avg}$ was covered in [1-3], and the estimation of gate count (or simply, area) $\mathcal{A}$ was explored in [4], where the problem was addressed using the notion of *average cube complexity* of a Boolean function. However, we have found that this area model [4] severely under-estimates the area on a class of circuits, as summarized in Table 1. In this paper, we propose a new model for predicting the area $\mathcal{A}$, leading to improvement in both the area prediction accuracy and run times relative to [4], as shown in Table 1. In our experiments, we have compared our estimated gate-counts to the gate-count for circuit implementations that were obtained using the SIS synthesis system, synthesizing for minimum area. As was the case with [4], our new model is still limited to single-output Boolean functions; we are in the process of generalizing it to multiple-output functions.

TABLE 1.
Area model comparisons with [4]

Circuit	Area SIS	Area new	Area [4]	CPU sec new	CPU sec [4]
s13207_o605	153	141	53	6	610
s13207_o604	147	156	51	7	605
s13207_o598	153	118	41	7	610
s13207_o597	136	143	41	7	580
s13207_o599	153	119	41	6	595
s9234_o42	186	184	20	81	580
s9234_o41	203	238	20	118	591

The proposed area model explores the structure of the Boolean space of the function, and computes a *complexity measure* for the Boolean function that we will show correlates well with its area. This is done by examining the essential prime implicants of the function. Based on the distribution of the essential primes (of both the onset and the offset) we propose two complexity measures called *linear measure* and *exponential measure*. These two measures are used to predict the area of the function.

The computation of essential primes is done using *espresso* [8]. For single output functions, which is our focus right now, this can be done quite efficiently. This is true even for large circuits because single output functions extracted from multiple-output specifications of large circuits turn out to be reasonably small Boolean functions. It was observed that the average time to run *espresso* for several hundred single-output functions extracted from ISCAS-89 and MCNC benchmark circuits was 0.876 sec, and the worst-case time observed was 16.53 sec. The only case where *espresso* tends to become expensive is for circuits composed of arrays of exclusive-or gates. These circuits were also problematic in [4], and they are also the source of problems for other CAD areas, such as BDD construction for verification. One way around this limitation is to require that the Boolean function specification explicitly list exclusive-or gates. In that case, these can be identified up-front and excluded from the analysis, so that the proposed method is applied only to the remaining circuitry. In any case, in the remainder of this paper we will not consider circuits composed of large exclusive-or arrays.

Before leaving this section, we should mention some previous work on layout area estimation from an RTL view. Wu et. al. [5] proposed a layout area model for datapath and control for two commonly used layout architectures based on the transistor count. For datapath units, the average transistor count was obtained by averaging the number of transistors over different implementations and, for control logic, they calculate the number of transistors from the sum of products (SOP) expression for the next state and control signals. A similar model was proposed by Kurdahi et.al. [6]. Both these models consider the effect of interconnect on the overall area, while [6] considers the effect of cell placement on the overall area. Since the controller area, in [5][6], is estimated based on the number of AND and OR gates required to implement the SOP expression, the optimal number of gates required to implement the function can be much smaller than the above sum. This is because it is frequently possible to apply logic optimization algorithms to give a much better implementation.

2. The Weighting Function

We start with a precise statement of the problem to be addressed. Consider an n-input single-output Boolean function $f(X)$. Given a target gate library, we would like to estimate the *minimum* number of gates ($\mathcal{A}$) required to implement the function, given only its high level description (Boolean equations), without performing any logic synthesis on the function $f(X)$.

The "sizes" of the essential prime implicants of the on and off-sets may give us a hint as to the complexity of the function at hand. By size of a cube we mean the number of literals in the cube. Thus, the size of the cube $c = x_1\bar{x}_2x_4$ is 3. We will study the complexity of the function by characterizing the distribution of the sizes of its essential prime implicants, as follows. The complexity of the on-set of a Boolean function f will be captured by the following measure:

$$\mathcal{C}_1(f) = \sum_{i=1}^{N} g(c_i)p_i \tag{2}$$

Here, $\mathcal{C}_1(f)$ is the complexity measure associated with the on-set of f, N is the number of distinct sizes of essential prime-implicants of the on-set of f, the set of integers $\{c_1, c_2, \ldots, c_N\}$ consists of the distinct sizes of the essential primes of the on-set, $g(.)$ is a monotonically increasing function, and p_i is a weight associated with the class of essential primes of size c_i. The complexity of the off-set will be similarly captured by $\mathcal{C}_0(f) = \mathcal{C}_1(\bar{f})$. Thus, in the following, it will be enough to discuss $\mathcal{C}_1(f)$.

For $\mathcal{C}_1(f)$, let the c_i be ordered such that $c_1 > c_2 > \cdots > c_N$. Let f_i refer to a Boolean subfunction of the original function f, defined so that its on-set consists only of the essential primes of sizes $c_1, c_2, \ldots, c_i$, where $1 \le i \le N$. With each min-term of the Boolean space, we associate a value of probability, so that all min-terms are equi-probable and the sum of their probabilities is equal to 1. We define the weight p_i as follows:

$$p_i = \begin{cases} \mathcal{P}(f_i) - \mathcal{P}(f_{i-1}), & \text{if } i > 1; \\ \mathcal{P}(f_1), & \text{if } i = 1. \end{cases} \tag{3}$$

where $\mathcal{P}()$ denotes probability. Thus, p_i is the probability of the set of all min-terms in the on-set of f that are covered by essential primes of size c_i, but not by essential primes of any larger size. With p_i thus defined, as probabilities, the expression (2) becomes equal to the *mean* of $g(.)$ (when $g(.)$ is assumed to take the value 0 with probability $1 - \mathcal{P}(f)$), $\mathcal{C}_1(f) = E[g(\mathbf{c})]$, and can be easily computed using Monte Carlo mean estimation techniques [7]. Using similar development, $\mathcal{C}_0(f)$ can also be computed using Monte Carlo simulation.

3. Linear and Exponential Measures

In this section we will define two specific complexity measures called the *linear measure*, denoted $\mathcal{L}(f)$, and *exponential measure*, denoted $\mathcal{E}(f)$, by making specific choices for the function $g(.)$. Let us first consider the *linear measure*. We will characterize the on-set of f with a linear complexity measure, based on (2), denoted by $\mathcal{L}_1(f)$ (and the off-set with $\mathcal{L}_0(f)$) by choosing $g(c)$ to be a linear function in cube size c, specifically, $g(c) = c$, so that:

$$\mathcal{L}_1(f) = \sum_{i=1}^{N} c_i p_i \tag{4}$$

We then define $\mathcal{L}(f)$ as follows:

$$\mathcal{L}(f) = \frac{\mathcal{L}_1(f) + \mathcal{L}_0(f)}{2} \tag{5}$$

Since cubes of larger sizes have a lower probability of occurrence, this measure is likely to be dominated by the cubes of lower sizes.

For the *exponential measure*, we characterize the on-set and off-set of the Boolean function with exponential complexity measures $\mathcal{E}_1(f)$ and $\mathcal{E}_0(f)$ respectively, based on (2), by choosing $g(c)$ to be an exponential function in cube size c, as follows:

$$\mathcal{E}_1(f) = \sum_{i=1}^{N} 2^{c_i} p_i \tag{6}$$

We then define $\mathcal{E}(f)$ as follows:

$$\mathcal{E}(f) = \log\Big\{ \min\{\mathcal{E}_1(f), \mathcal{E}_0(f)\} \Big\} \tag{7}$$

where log is logarithm to base 2. This complexity measure compensates for reduction in probability of occurrence in larger sized cubes by associating them with larger measure. Hence it is likely to be dominated by cubes of larger sizes.

We have observed that the presence of essential cubes of size one (cubes consisting of a single literal) can adversely affect the accuracy of the area estimation. This is because these cubes have a negligible effect on the gate count (a single OR gate) but have a big effect on the output probability value. Their presence also skews the probability distributions and makes the Monte Carlo estimation much more expensive. We have found that the best practical method for accounting for these cubes is to in effect exclude them from the summation (2) used to compute $\mathcal{C}_1(f)$, and similarly for $\mathcal{C}_0(f)$. This leads to improved estimation speed and much improved accuracy. Thus, the results to be presented below make this modification to the essential cube distribution before carrying out the area prediction.

4. The Area Model

For a given n, consider the set of all Boolean functions on n inputs *and* whose output probability is $\mathcal{P}(f) = p$, based on all inputs being independent and with 0.5 probability. For a number of randomly generated Boolean functions from this set, we computed $\mathcal{L}(f)$ and $\mathcal{E}(f)$ using our algorithm and obtained an estimate of the gate-count $\mathcal{A}(f)$ from an optimized implementation of the function using SIS (by *randomly generated*, we mean that these functions were selected by making a random choice for each point in the Boolean space, as to whether it belongs in the on-set or off-set of the function). The curves of $\mathcal{A}(f)$ versus $\mathcal{L}(f)$ and $\mathcal{E}(f)$ are close to exponential. We have found that not only do randomly generated Boolean functions fall on these almost-exponential curves, but also typical VLSI functions fall on it or close to it. A similar observation was also made in [4]. Thus, these curves of $\mathcal{A}$ versus $\mathcal{L}$ and $\mathcal{E}$ are very important and are in fact the essence of our area prediction model.

We generate a family of such curves capturing the variation of $\mathcal{A}$ with $\mathcal{L}$ and $\mathcal{E}$ for various values of the output probability $\mathcal{P}(f)$. Hence, given the output probability of the Boolean function and $\mathcal{L}(f)$, we use the curve corresponding to that value of probability and predict $\mathcal{A}_l(f)$. The procedure for computing $\mathcal{A}_e(f)$ is similar. Note that these curves need to be generated only once, which is an up-front once-only cost, and they can then be used to predict the area of various functions. An important consideration is what the largest n should be for which these curves need to be generated. Please refer to [4] for detailed discussion on this issue, as an approach similar to [4] can be adopted.

We empirically observed that in many cases $\mathcal{A}_l(f) \leq \mathcal{A}_e(f)$. Also, we observed that the average of $\mathcal{A}_l(f)$ and $\mathcal{A}_e(f)$ was better correlated with $\mathcal{A}(f)$ than either $\mathcal{A}_l(f)$ or $\mathcal{A}_e(f)$. This relates to the comments made in section 3, namely that the linear model is mostly affected by small cubes while the exponential model is dominated by large cubes; the average of the two performs better than either of them acting separately. Hence we measure the area complexity $\mathcal{A}(f)$ as an average of $\mathcal{A}_l(f)$ and $\mathcal{A}_e(f)$. The area complexity as measured by this *average area model* is given by:

$$\mathcal{A}_a(f) = \frac{\mathcal{A}_l(f) + \mathcal{A}_e(f)}{2} \tag{8}$$

In the next section we present empirical results showing the performance of the *average area* model on several benchmarks. Also, we compare this model with $\mathcal{A}_e(f)$, and show that this model performs better on the average.

5. Empirical Results

Before presenting the actual data, a word on how the benchmarks were generated is in order. We used single output functions extracted from the ISCAS-89 and MCNC benchmark suite. These single output functions were optimized using SIS, for minimum area using *rugged.script* and mapped using *lib2.genlib*.

The area complexity values (gate-count predictions) of these benchmarks, using our model, were computed as follows. Firstly, the probability of the Boolean function was estimated to a 5% error tolerance and 95% confidence using a Monte-Carlo approach. The complexity measures $\mathcal{L}$ and $\mathcal{E}$ were estimated to an accuracy of 90% with a confidence of 90% on a SUN sparc-5 workstation after the essential primes were computed (using *espresso*). The average run time for all the benchmarks was about 1.9 cpu seconds and worst case run time was 85 cpu seconds. Hence, the total time required for computing the complexity measures was on the average about 4 cpu seconds and the worst case value observed was 119 cpu seconds. The above computed complexity measures were used along with output probability $\mathcal{P}(f)$ to estimate $\mathcal{A}_l(f)$ and $\mathcal{A}_e(f)$, which in turn were used to compute $\mathcal{A}_a(f)$.

The comparison between the SIS-optimized gate count values and the predicted gate counts given by $\mathcal{A}_a(f)$, $\mathcal{A}_e(f)$ and $\mathcal{A}_l(f)$ are given in Figs. 1, 2 and 3 respectively. A blow-up of the plot in Fig. 1 is also given in Fig. 4. The performance of these models is much better than the area model in [4] in terms of accuracy and run-times.

6. Conclusions

In this paper, we have proposed an area model for predicting the area of single-output Boolean functions, that is superior to the model in [4], both in accuracy of prediction and run time. Moreover, the feasibility of the proposed model was demonstrated on a realistic library, *lib2.genlib* of the SIS library suite. The improvement in prediction accuracy was achieved through the definition of two new complexity measures, *linear measure* and *exponential measure*, which depend on the distribution of essential prime implicants of the function at hand. It was also empirically demonstrated that the *average area* model has a better correlation with SIS-optimized area than the areas predicted from the *linear measure* and *exponential measure*. Like [4], this work relates structural attribute of the function (area) to its functional attribute (complexity measures), which is a definite requirement for high-level power estimation. We are currently working on extending this model to multi-output functions.

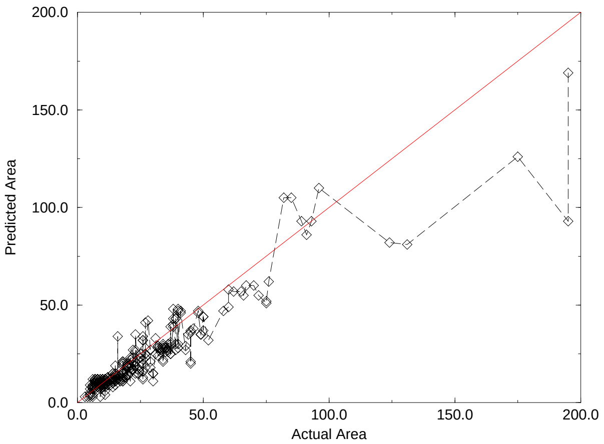

Figure 1. Actual versus Predicted Area for *average area* model.

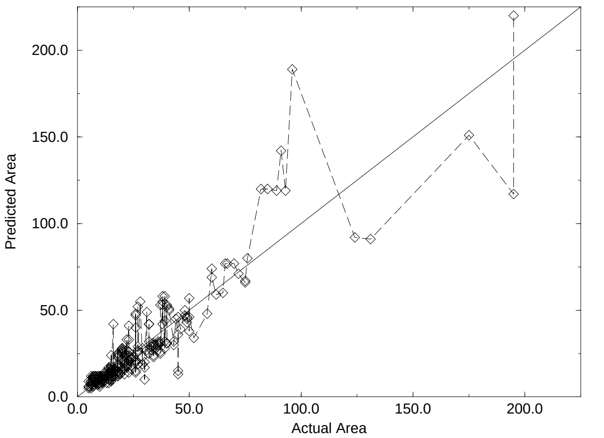

Figure 2. Actual versus Predicted Area for *Exponential measure*.

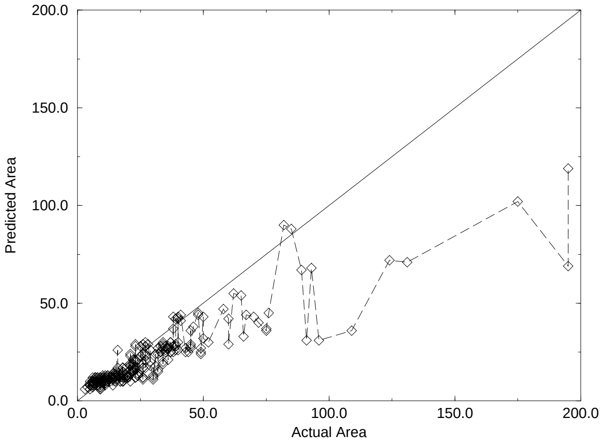

Figure 3. Actual versus Predicted Area for *Linear Measure*.

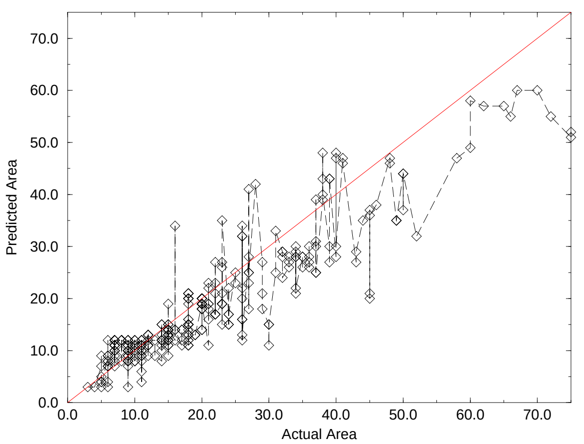

Figure 4. Actual versus Predicted Area for *average area* model.

References

[1] F. Najm, "Towards a high-level power estimation capability," *ACM/IEEE International Symposium on Low-Power Design*, pp. 87–92, 1995.

[2] D. Marculescu, R. Marculescu and M. Pedram, "Information theoretic measures of energy consumption at register transfer level," *International Symposium of Low Power Design*, pp. 81-86, 1995.

[3] M. Nemani and F. Najm, "Towards a high-level power estimation capability," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 15, no. 6, pp. 588-589, June 1996.

[4] M. Nemani and F. Najm, "High-level power estimation and the area complexity of Boolean functions,"*International Symposium of Low Power Electronics and Design*, pp. 329-334, 1996.

[5] A. C-H. Wu, V. Chaiyakul and D. D. Gajski, "Layout area models for high level synthesis," *International Conference on Computer Aided Design*, pp. 34-37, 1991.

[6] F. J. Kurdahi, D. D. Gajski, C. Ramachandran and V. Chaiyakul, "Linking register transfer and physical levels of design,"*IEICE Transactions on Information and Systems*, September 1993.

[7] M. Xakellis and F. Najm, "Statistical estimation of the switching activity in digital circuits," *31st ACM/IEEE Design Automation Conference*, pp. 728–733, 1994.

[8] R. K. Brayton, G. D. Hatchel, C. T. McMullen and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Boston, MA: Kluwer Academic Publishers, 1984.