High-Level Area Estimation*

Kavel M. Büyükşahin
ECE Dept. and Coordinated Science Lab.
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801, USA
buyuksah@uiuc.edu

Farid N. Najm
ECE Department
University of Toronto
Toronto, Ontario, Canada M5S 3G4
f.najm@utoronto.ca

ABSTRACT

Early power estimation requires one to estimate the area (gate count) of a design from a high-level description. We propose a method to do this that makes use of the concept of Boolean networks (BN) and introduces an invariant area complexity measure which captures the gate-count requirement of a design. The method can be adapted to be used at different points on the area/delay tradeoff curve, with different synthesizer/mapper tools, and different target gate libraries. The area model is experimentally verified and tested using a number of ISCAS and MCNC benchmark circuits and two different target cell libraries, on two different synthesis systems.

Categories and Subject Descriptors

B.5.2 [**RTL Implementation**]: Design aids

General Terms

Design

Keywords

Area estimation, Boolean networks

1. INTRODUCTION

The rapid increase in design complexity and reduction in design time has resulted in the need for CAD tools that can help make design decisions early in the design process. Area (roughly, the number of gates required to implement a Boolean function) and power dissipation are two of the most important criteria that have to be taken into account while making these decisions. However, to be able make these decisions early, there is a need for methods to estimate the area and power consumption from a design description of the circuit at a high level of abstraction. Some techniques have been proposed to estimate the area and the power consumption of a circuit given a *register transfer level* (RTL) description [6, 2, 10, 11, 4]. In [6, 2], the authors propose a method to estimate power and area given only a functional view of the design, such as the Boolean equations. To do this, they introduce an area model based on the number and sizes of the prime implicants of the function. Practical use of this technique is limited, however, as finding all the prime implicants of a function can be computationally expensive. In [10], the authors build an area and delay model which uses actual design information obtained by implementing a subset of the design, and using this model for the whole circuit. Although the method's accuracy is quite good (when the "correct" subset is chosen), the requirement that the whole circuit be in a technology-independent minimal form, and the problem of choosing the subset to implement make it unattractive for practical use. The methods proposed in [11] and [4] both make use of the SOP representation of a function, and estimate the area based on the total number of AND and OR gates required in this representation. Typically, the actual number of gates required will be much smaller than this number after optimization.

In this paper, we propose a method to estimate the area (based on gate count) of a design which is described at a high-level of abstraction. Specifically, we propose an area estimation capability, using only the Boolean network representation (to be defined later in the paper) of the function, and a primitive-independent *area complexity measure* extracted from this representation.

2. METHODOLOGY

Our method of estimating the gate count consists of three steps. First, we build a Boolean network representation of the given design. Then, we extract the relevant parameters of this network to compute the area complexity measure. In the last step, using the area complexity measure, we get an estimate for the gate count of the design based on a previously-characterized target gate library and a synthesis tool. In the following subsections, we will look into each of these steps in detail.

2.1 Boolean Networks

A Boolean network (BN) is a directed acyclic graph representing a set of Boolean equations. Each node in the BN corresponds to a Boolean primitive, and the edges correspond to the connections between these primitives [1]. Building a BN corresponding to a design is easy once the set of primi-

*This project was supported in part by the Semiconductor Research Corporation (SRC 99-TJ-682), with funds from IBM Corporation.

ISLPED'02, August 12-14, 2002, Monterey, California, USA.

tives to be used as nodes is chosen. It simply involves translating the given format into a pseudo gate-level format where each gate corresponds to a Boolean primitive (such as OR, AND, NOR, NAND, XOR, NOT, etc).

Obviously, one can build many different Boolean networks for a given set of Boolean equations, using different primitives as nodes of the network. Thus, the BN not being canonical, it would not seem to be a good means to asess the computational cost implicit in a Boolean function. Nevertheless, one would also expect the different BN representations of the same function to have some invariant properties as they are representing the same Boolean functionality. This was the motivation for our work, finding an invariant attribute of a BN that is representative of the function which can be mapped easily to an estimate of the final gate count that that function would require when synthesized to a given gate library.

2.2 Complexity measure

There is a lot of work in the literature that addresses the complexity of Boolean functions [9, 5, 7, 3]. In many cases, the complexity measure of a Boolean function has been expressed in terms of the function's output entropy and an exponential in the number of nodes. It has also been observed [6] that many of these measures break down in practice, hence the need for more practical, efficient, techniques for assessing complexity and relating it to a gate count estimate.

In our work, a BN is simply a graph. Given any graph, there are many parameters that can be extracted, such as the number of nodes, number of edges, average in-degree (fan-in), average out-degree (fan-out), depth, size of cutsets, topological order of the nodes, minimum spanning trees, etc. Therefore, we can talk about the number of nodes, average fan-in, average fan-out, or depth of a BN.

Our gate count estimation method is based on an area complexity measure extracted from a BN that represents the given design. To be useful, this measure should satisfy two conditions: *(i)* It should be invariant among the different BN representations of the same design. The original design may be in any of the various formats, and users might want to build the BN using any set of primitives that is convenient for them. These will result in very different BN representations of the same function. We want our area complexity measure to be constant for all those representations, at least with some approximation. *(ii)* Obviously, the area complexity measure should have a well-defined relationship with the optimized gate count of the final circuit in the target gate library. Furthermore, this "model" should be applicable for different cell libraries, or different synthesizer/mapper tools.

We have found that one complexity measure satisfying these conditions is given by:

$$C(B) = n \cdot \overline{f_{in}} \cdot \overline{f_{out}} \tag{1}$$

where $C(B)$ is the area complexity measure extracted from Boolean network B, n is the number of nodes in B, $\overline{f_{in}}$ is the average fan-in (in-degree), and $\overline{f_{out}}$ is the average fan-out (out-degree), of the nodes of B. We can re-write this complexity measure as:

$$C(B) = \overline{f_{in}} \cdot E_{out}$$

where $E_{out} = n \cdot \overline{f_{out}}$ is the total number of out-going edges.

We have found that this complexity measure is approximately invariant for different BNs of the same function. One reason for this is as follows. Notice that f_{in} for a BN node represents a rough measure of gate count, or silicon cost required to implement that node. One can also think of f_{in} for a BN node as a first-order measure of the "computational work" being performed at that node. Thus, $\overline{f_{in}}$ is the average computational work per node in the BN. On the other hand, E_{out}, which is the total number of all out-going edges, is a measure of the connectedness of the BN. It is, in fact, a measure of the overall "communication cost" inside the BN.

If $\overline{f_{in}}$ is somehow increased (due to the use of a different set of primitives), then more of the overall computation would be done inside the BN nodes themselves and there would be less overall communication that is needed between the nodes. Hence, as $\overline{f_{in}}$ is increased (decreased), E_{out} should decrease (increase). Our experiments verify this claim, and actually indicate a very simple relationship between the two: $\overline{f_{in}} \cdot E_{out} \approx$ constant.

The preceding argument also helps explain why this constant can be used as an area complexity measure. Since the constant value combines the cost of computation and communication, it can be viewed as the overall computational work of the Boolean function, and it can be used as a measure of its complexity, and ultimately, gate count requirements. The next section explains how this can be done.

2.3 Gate count estimation

The preceding two sections explained how we can get an area complexity measure given a high-level (Boolean) description of a design. All the steps taken to do that were independent of the target gate library and the synthesis tool that will be used to synthesize/map this design. The third step in our estimation process relates this (library independent) complexity measure to the actual gate count of the optimal circuit in a given target gate library.

This step can be explained in two phases. The first phase is the characterization phase. In this phase, we find the relationship between the area complexity measure and the actual gate count obtained by optimizing, synthesizing, and mapping the function to a target gate library using a synthesizer/mapper. Once the tool and the target gate library are characterized, we can use this relationship to estimate the gate count requirements of *other* designs without going through the optimization-synthesis-mapping process. We were able to find a very simple and well-defined relationship between the complexity measure and the gate-count requirement for two different synthesis tools. The details will be explained in the experimental results section.

3. EXPERIMENTAL RESULTS

We will demonstrate the invariance of the complexity measure, as well as discuss tunability and verification of the model.

3.1 Verification of the area complexity model

We will show that the proposed complexity measure is invariant for different BN representations of the same circuit and that it has a well defined relationship with the optimized gate count of the synthesized circuit. We will also show that this complexity measure performs better than a simpler measure such as node count.

To obtain different BNs for the same function, we have used different sets of primitive Boolean nodes and have built five different BNs from each of the benchmark circuits, and extracted the relevant parameters. Using these parameters, we have computed the area complexity measure as defined in (1). The results are shown in Fig. 1 for a number of ISCAS and MCNC benchmark circuits. The primitive sets we have used consist of inverters, OR and AND gates with various support set sizes. The set OR2 consists of inverters and 2-input OR gates. For OR4, we have added 4-input OR gates to the primitive set, and so on. As can be seen from the figure, the area complexity measure for a design is approximately constant across different BNs built with different Boolean primitives.

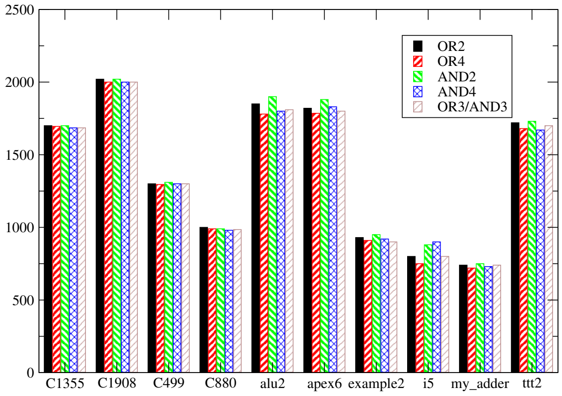


Figure 1. Complexity measure for some designs under different sets of primitives.

To show that our complexity measure performs better than the node count, we show a bar chart of the node count for different BNs, in Fig. 2. As can be seen from the figure, the node counts obtained by using different primitive sets vary substantially. This makes node count a poor choice as a complexity measure, because it varies with variations in the structure of the BN even though the Boolean function itself is not changing.

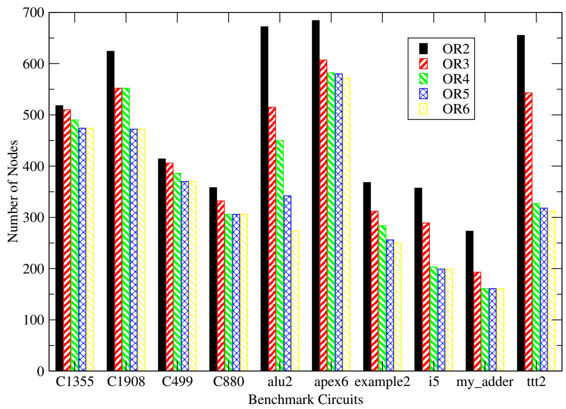
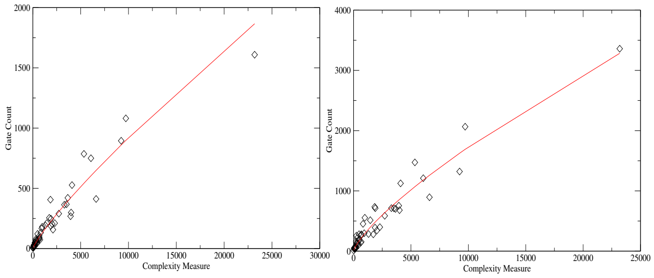


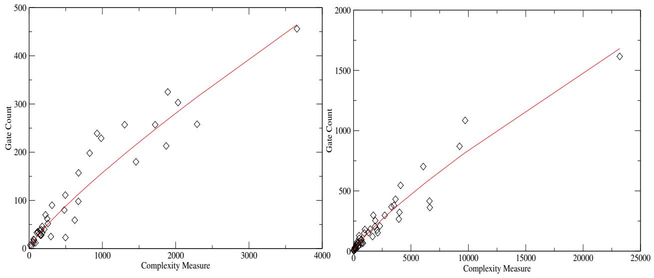
Figure 2. Node count of BNs for different sets of primitives.

In order to study the relationship between our complexity measure and the gate count of the optimized circuit, we have built BNs for a number of benchmark circuits and extracted the area complexity measure from them. We optimized these circuits for minimum area using Synopsys Design Compiler (DC) and mapped the optimized circuits to cell library (the *class* library that comes with DC). The correlation plot in Fig. 3a shows a clear relationship with the optimized gate count of the synthesized circuit. The relationship is a simple power law of the form $y = A \cdot x^B$.



(a) **(b)**

Figure 3. Gate count vs. Complexity measure for Synopsys DC and *class* library
(a) at minimum area point (b) at minimum delay point



(a) **(b)**

Figure 4. Gate count vs. Complexity measure, (a) for SIS and the *class* library, and (b) for Synopsys DC and the *Odyssey* library.

In the next section, we will show that this model can be tuned for different synthesis tools, different target libraries, and different points on the delay/area trade-off curve simply by re-calculating the model parameters A and B, without any change to the complexity measure.

3.2 Verification of the tunability of the model

To be practically usable, the model should be tunable with respect to the synthesis tool, the target library, and the delay specification. To begin with, we will show that circuits that are optimized for points other than the minimum area point still have the same well-defined relationship, only with different model parameters. Fig. 3b is a plot showing the relationship between the gate count at the minimum delay point and the area complexity measure. The model is of the form $y = A \cdot x^B$ with $A = 1.5153$ and $B = 0.7642$.

To show that our model is tunable with respect to the synthesis tool, we have generated a plot similar to Fig. 3a using SIS [8]. We have used the *script.rugged* of SIS to optimize the benchmark circuits for minimum area, and mapped them on the same target library (*class*). Fig. 4a shows that our model is still valid with a different synthesis tool. The model parameters in this case are $A = 0.4911$ and $B = 0.8352$.

Finally, and as for tunability of the model for different target libraries, we have optimized and mapped the benchmark circuits using Synopsys DC and the Odyssey cell library for TSMC 0.25μ technology. The correlation plot is shown in Fig. 4b, and the model parameters are $A = 0.4000$ and $B = 0.8301$.

3.3 Testing the model

To test our model, we have used a number of ISCAS and MCNC benchmark circuits. These circuits were synthesized and mapped on two different target gate libraries using SIS, and Synopsys DC. The set of primitives used for building the BNs were inverters and 2-input OR gates.

For the characterization step, we randomly chose a number of circuits, and built the BN for each of them. Then we extracted the number of nodes, average fan-in, and average fan-out from these BNs. We computed the area complexity measure using (1) with these parameters. We then optimized the circuits using the desired tool and the target library and extracted the gate counts of the optimized circuits. Performing regression analysis on these data, we computed the model parameters A and B, and hence characterized the tool/library/delay point. We then estimated the optimized gate count for other benchmark circuits using the model obtained by the characterization step and compared these estimates with the optimized gate counts. Fig. 5 shows the characterization step for Synopsys DC and *class* library at the minimum area point. The values of A and B are 0.3664 and 0.8457 respectively.

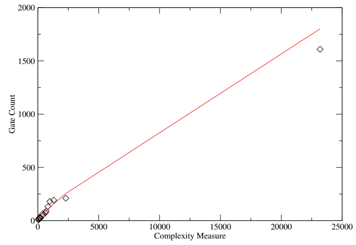


Figure 5. The characterization step for DC and the *class* library at min area point.

Fig. 6 shows the estimated gate count vs. actual gate count for Synopsys DC and the *class* library at the minimum area point. The average error in our estimation is 24.5%. We have observed an average estimation error of 23.3% using SIS and the *class* library, and 25.3% for Synopsys DC with *Odyssey* cell library. Due to space limitations, we cannot include the correlation plots for these data.

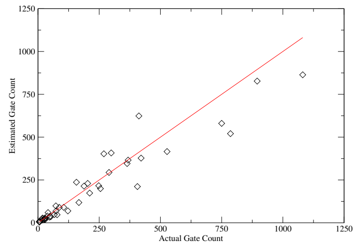


Figure 6. Estimated vs. actual gate count for Synopsys DC and the *class* library.

3.4 Limitations

The proposed complexity measure has some limitations. For instance, if a BN contains significant redundant logic, such as when large parts of the function are redundant and would be removed if synthesized, then it is clear that a simply structural complexity measure as we have proposed will over-estimate the gate count. One point to be made in this regard is that it is good that the estimation is conservative in this case. Another point to be made is that we did not encounter this behavior in any of the test cases that we looked at, and thus we are inclined to think that this measure has some virtue.

Another point worth making relates to the use of this complexity measure as a way to predict the requirements of a soft IP (intellectual property) block. If one is to make available a synthesizable description of a large design, as soft IP, it is hardly likely that they will package this IP in a way that includes large redundant logic portions. Thus, it seems likely that the proposed complexity measure would be very useful to the end user in this case.

4. CONCLUSION

In this paper, we have presented a method for estimating the gate-count (area) requirement of a combinational circuit given a high-level design description. This method makes use of the Boolean network concept, and defines an area complexity measure which is invariant across the different BN representations of the same design. The method can be tuned for different synthesizer/mappers, target gate libraries and delay specifications. It is shown to be very robust with respect to changes in the BN primitives.

5. REFERENCES

[1] K. A. Bartlett and et. al. Multilevel logic minimization using implicit don't cares. *IEEE Transactions on Computer-Aided Design*, 7(6):723–740, June 1988.

[2] K. M. Buyuksahin and F. N. Najm. High-level power estimation with interconnect effects. In *Proc. International Symposium on Low Power Electronics and Design (ISLPED)*, pages 197–202, Italy, July 2000.

[3] K.-T. Cheng and V. Agrawal. An entropy measure for the complexity of multi-output Boolean functions. In *Proc. ACM/IEEE Design Automation Conference*, pages 302–305, 1990.

[4] F. J. Kurdahi and et. al. Linking register-transfer and physical levels of design. *IEICE Transactions on Information and Systems*, 76(9):991–1002, 1993.

[5] D. E. Muller. Complexity in electronic switching circuits. *IRE Trans. on Electronic Computers*, 5(1):15–19, 1956.

[6] M. Nemani and F. N. Najm. High-level area and power estimation for VLSI circuits. *IEEE Transactions on Computer-Aided Design*, 18(6):697–713, June 1999.

[7] N. Pippenger. Information theory and the complexity of Boolean functions. *Mathematical Systems Theory*, 10(1):129–167, 1977.

[8] E. M. Sentovich and et. al. SIS: A system for sequential circuit synthesis. Memorandum UCB/ERL M92/41, Electronics Research Laboratory, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720, May 1992.

[9] C. E. Shannon. The synthesis of two-terminal switching circuits. *Bell System Technical Journal*, 28(1):59–98, 1949.

[10] A. Srinivasan and et. al. Accurate area and delay estimation from RTL descriptions. *IEEE Transactions on VLSI Systems*, 6(1):168–172, Mar. 1998.

[11] A. C.-H. Wu and et. al. Layout-area models for high-level synthesis. In *Proc. International Conf. Computer-Aided Design (ICCAD)*, pages 34–37, 1991.