Early Power Estimation for VLSI Circuits

Kavel M. Büyükşahin, *Member, IEEE,* Farid N. Najm, *Fellow, IEEE*

***Abstract*— Early power estimation, a requirement for design exploration early in the design phase, must often be done based on a design specification that is available only at a high level of abstraction. One way of doing this is to use high-level estimation of circuit total capacitance and average activity. This paper addresses these problems and proposes a high-level area estimation technique based on the complexity of a Boolean network representation of the design. In addition to the high-level area estimation, the paper also proposes a high-level activity estimation methodology that is capable of handling correlated input streams. High-level power estimates based on the total capacitance and average activity estimates are also given.**

***Index Terms*— Power estimation, Boolean networks.**

The increasing complexity and high-performance requirements of modern integrated circuits have naturally led to very high power consumption. Concern with the power consumption problem is at such a high level that, instead of speed and density, power profile is fast becoming the major consideration in high-end microprocessors [1]. Power is no longer a secondary issue, to be considered only after area/delay trade-offs are performed. It is very important to be able to design low-power VLSI circuits that are small, and fast. The two dimensional design exploration space is a thing of the past, the designer has to keep in mind that she has three objectives: low-power, high performance, and small layout area. As a result, there is an urgent need for high-level power estimation and optimization.

One may employ two types of modeling approaches for the power estimation problem: bottom-up and top-down. Bottom-up techniques attempt to measure the power dissipated by existing implementations and produce a model based on the measurements. This "power macro-model" of a block can be used during high-level power estimation without performing more expensive, lower-level power estimation on it. Examples of bottom-up approach are the power factor approximation (PFA) of [2], the dual bit type (DBT) model of [3], and the table based approach of [4]. Bottom-up approach is best suited for library based designs (where the library includes relatively complex blocks). However, in practice, completely library based designs are very rare. In every design, there will be application specific portions that have not been previously implemented. For those portions, one needs a power model that works on a high-level description of the design and does not need a low-level implementation to be available – a top-down model. This paper proposes one such technique, in which the power is estimated as the product of an estimate of total capacitance and an estimate of average switching activity. Total capacitance estimation requires one to compute a measure of circuit *area*, typically gate count.

More specifically, we propose a top-down power estimation technique that works at the structural register-transfer level (RTL), where memory elements (flops or latches) are well defined, but the combinational logic is specified simply as Boolean equations. As the number of latches/flops is known at this level, it is possible to estimate the power consumption of the memory elements with the help of bottom-up power macromodels. Therefore, the problem of power estimation at the structural RTL reduces simply to estimating the power of the combinational parts, hence the gate count and the activity of the combinational parts. In this paper, we will employ an approach similar to the technique introduced in [5] to estimate the power consumption of the combinational parts of a design at the structural RTL. In [5], authors introduce a complexity based area estimation technique, an average capacitance estimation technique, and a high-level activity estimator. In our work, we propose a new area estimation technique, which is considerably less computationally expensive than the one introduced in [5], and improve on the activity estimator by adding the capability to handle spatially correlated input streams.

The organization of this paper is as follows: In Chapter I we will introduce the high-level area estimation approach used in this work. In II, we propose an area estimation technique that, combined with the capacitance estimation technique of Chapter IV [5], will let us estimate the total capacitance of a circuit from a high-level view. In V, we will summarize the activity estimation technique introduced in [6], and in Chapter VI we will introduce our improvements on this method to handle spatially correlated input streams. In VII, we combine the results of our capacitance estimator and the activity estimator to come up with high-level power estimates. We end the paper with some conclusions presented in VIII.

I. Background

In this section, we will introduce an overview of the high-level power estimation methodology. This issue was previously addressed by Nemani and Najm in [6], [5].

In this work, we restrict ourselves to the static fully complementary CMOS technology. For a combinational logic circuit composed of N logic gates, whose output nodes are denoted $x_i, i = 1, 2, \ldots, N$, we can write the average power consumed by

$$P_{avg} = \frac{1}{2} V_{dd}^2 \sum_{i=1}^{N} C_i D(x_i) \tag{1}$$

where C_i is the total capacitance at the node i, and $D(x_i)$ is the transition density [7] of node x_i (average number of logic transitions per second).

At the RT level, combinational logic is described as a Boolean network whose nodes implement an arbitrary Boolean

This research was supported in-part by the Semiconductor Research Corporation (SRC), under contract 682.001.

function. No internal details of the circuit are known at his level. Therefore, approximations to the gate-level power model are inevitable for deriving a power model at the RT level. The model proposed under this constraint is

$$P_{avg} \propto \mathcal{C}_{tot} \times \mathcal{D}_{avg} \tag{2}$$

where $\mathcal{D}_{avg}$ is a measure of the average node switching activity and $\mathcal{C}_{tot}$ is total circuit capacitance.

II. Area Estimation

In this section, we address the problem of estimating the area (gate count) of a Boolean function from its high-level description (Boolean equations) only. The area estimate will then be used to obtain a measure of the circuit *total capacitance*, which will then be multiplied by the average activity to estimate the power. A preliminary version of the material in this section was presented at [8].

The problem of estimating the total capacitance of a circuit from only a high-level description is an involved one. The circuit capacitance depends very heavily on the gate library and the circuit speed. A given Boolean function can have many different implementations resulting in different total capacitance values. Nevertheless, to be able to predict the power consumption of a circuit at the RT-level, one needs a measure of the total capacitance of the circuit. In this work, we use the following approximation based on the work of [5] for the total capacitance

$$\mathcal{C}_{tot} = \mathcal{A}\mathcal{C}_g \tag{3}$$

where $\mathcal{C}_{tot}$ is the total capacitance associated with the Boolean function, $\mathcal{A}$ is the total number of gates (or the gate count complexity) required for an optimal area implementation of the Boolean function given a technology-dependent library and a certain delay specification, and $\mathcal{C}_g$ is the average capacitance associated with a gate in the library under the given delay conditions. Given a technology library, the most difficult problem in obtaining a reasonable estimate of the total capacitance is the estimation of $\mathcal{A}$. We will refer to $\mathcal{A}$ as "area complexity" from here on.

In this work, we propose a method to estimate the area complexity (gate count) of a design described at a high-level of abstraction. Specifically, we propose an area estimation capability, given only the Boolean equations (or any other high-level description of the design) using only the Boolean network representation (to be defined later in the chapter) of the function, and a primitive-independent *area complexity measure* extracted from this representation.

Our method of estimating the gate count basically 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 computed in the previous step, we get an estimate for the gate count of the design in a previously-characterized target gate library and a synthesis tool, for the given delay constraints. In the following subsections, we will look into each of these steps in detail.

A. Boolean Networks

Definition 1 (Boolean Network) *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 [9].*

Building a BN corresponding to a design is easy once the set of primitives 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 asses 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.

B. Complexity Measure

There is a lot of work in the literature that addresses the complexity of Boolean functions [10], [11], [12], [13]. 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 [5] 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 cut-sets, 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 the 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.

C. Area Complexity Measure

We have found that a complexity measure satisfying both of the conditions in II-B to be

$$C(B) = n \cdot \overline{f_{in}} \cdot \overline{f_{out}} \tag{4}$$

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

$$C(B) = \overline{f_{in}} \cdot E_{out} \tag{5}$$

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:

$$\overline{f_{in}} \cdot E_{out} \approx \text{ constant} \tag{6}$$

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 (4). The results are shown in Table I for a number of ISCAS and MCNC benchmark circuits.. The primitive sets we have used consist of inverters and OR gates with various support set sizes. Under the column headed OR2 are the complexity measures computed from the BNs consisting of inverters and 2-input OR gates. For column OR3, we have added 3-input OR gates to the primitive set, and so on. The column heading "Simple" corresponds to a simple gate library containing an inverter, a NAND2, a NOR2 and an AND-OR-INVERT gate. The rightmost two columns display the mean and the standard deviation of these values. As can be seen from the table, the area complexity measure for a design is approximately constant across different BNs built with different Boolean 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 Figure 1. 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.

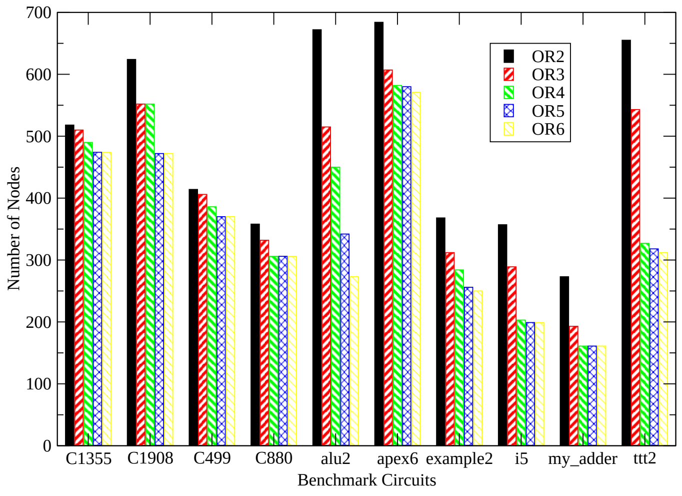

Fig. 1. Node count of BNs for different sets of primitives.

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 bundled with DC). The correlation plot in Figure 2 shows a clear relationship with the optimized gate count of the synthesized circuit. The relationship is a simple power law of the form

$$y = m \cdot x^n \tag{7}$$

where m and n are fitting coefficients.

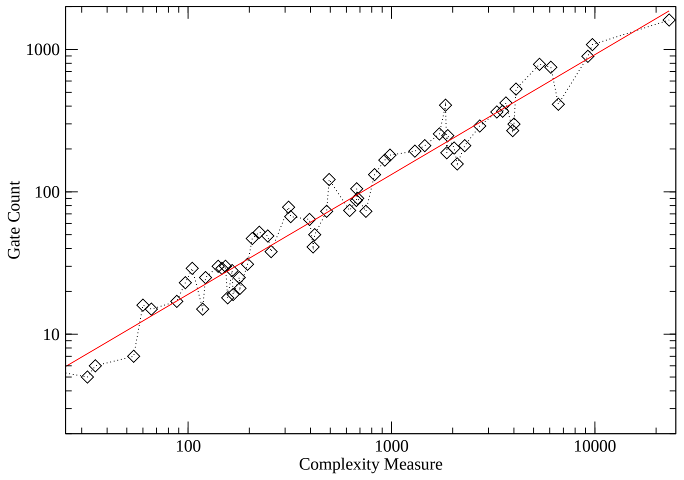

Fig. 2. Gate count - Synopsys DC, *class* library.

TABLE I

COMPLEXITY MEASURE FOR SOME DESIGNS UNDER DIFFERENT SETS OF PRIMITIVES.

Circuit	OR2	OR3	OR4	OR5	OR6	Simple	Mean	σ
C1355	1720	1717.36	1711.54	1707.75	1707.75	1736.7	1716.85	10.94
C1908	2032.54	2011.87	2011.87	2007.27	2007.27	2028.07	2016.48	10.99
C499	1304	1300.69	1293.26	1288.3	1288.3	1296.6	1295.19	6.46
C880	983.24	967.988	954.562	954.562	954.562	1003.28	969.70	20.01
alu2	1870.61	1793.97	1778.65	1794.96	1856.88	1787.79	1813.81	39.35
apex6	1843.42	1797.3	1784.08	1783.07	1778.59	1789.61	1796.01	24.10
example2	927.098	886.186	870.528	857.57	855.232	848.87	874.25	29.10
i5	826	771.176	714.483	712.462	712.462	743.89	746.75	45.34
my_adder	674.33	623.482	611.255	611.255	611.255	667.053	633.11	29.59
ttt2	1654.2	1577.93	1499.94	1500.75	1501.62	1484.84	1536.55	66.44

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.

D. Area Complexity Model

The preceding section 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 relate this library independent measure to the actual gate count of the optimal circuit in a given target gate library.

This step of the estimation process 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 both of the synthesis tools (SIS and Synopsys Design Compiler) that we experimented with.

The steps involved in the estimation process can be summarized as follows:

1) Characterization of the synthesizer/mapper and the gate library:
   - A number of previously implemented designs are chosen.
   - The BNs corresponding to these designs are built.
   - The area complexity measure as defined in (4) is computed.
   - The relationship between the complexity measure and the gate-count requirement is established (by regression analysis, or building look-up tables).
2) Gate-count estimation:
   - The BN for the new design is built.
   - Area complexity measure for the design is computed.
   - Using the relationship established in the characterization step, the actual gate count requirement of the new design is estimated.

Notice that the characterization step needs to be done only once for a given tool and a target gate library. The computational cost of this process depends heavily on the synthesis tool, synthesis script and the desired coverage (number and size of benchmark designs). As an example, one such characterization process involving 16 benchmark circuits ranging in size from 7 to 2000 gates ran for 2 CPU hours using Synopsys DC and high-effort script on a SUN UltraSparc 10 Workstation.

E. Delay Constraints

As we have mentioned earlier, the actual implementation of a Boolean function depends very much on the delay constraints. Therefore, our high-level area estimator needs to be able to handle user-defined delay constraints in order to be considered useful. In this section, we will show that it is possible to handle user-defined delay constraints (given in the form of *parameterized delay* in our area estimation model. For that, we first need to define what we mean by *parameterized delay*.

Definition 2 (parameterized delay, λ) *For a given Boolean function, let t_{min} be the delay of the minimum-delay gate-level implementation, and t_{max} be the delay of the maximum-delay (i.e., minimum area) implementation. The delay parameter λ corresponding to an arbitrary delay value t_d can be computed as*

$$\lambda = \frac{(t_d - t_{min})}{(t_{max} - t_{min})} \times 100 \qquad (8)$$

In other words, λ is the linear measure of the actual delay constraint relative to minimum ($\lambda = 0$) and maximum ($\lambda = 100$) delay points.

We will now demonstrate that the area complexity model holds true at not only the minimum delay point as shown in Figure 2, but also at other points on the delay-area trade-off curve. More specifically, we will demonstrate that a similar relationship can be observed between the area complexity measure and the actual area complexity (gate count) at minimum delay ($\lambda = 0$) and 50% delay ($\lambda = 50$) points as done at the minimum area ($\lambda = 100$) point.

Figures 3 and 4 show the relationship between the gate count and the area complexity measure at $\lambda = 0$ and $\lambda = 50$ respectively. As you can see, the relationship can still be modeled with a function of form $y = m \cdot x^n$ as in (7). The only difference we have observed for these three cases was the values taken by the model parameters m and n.

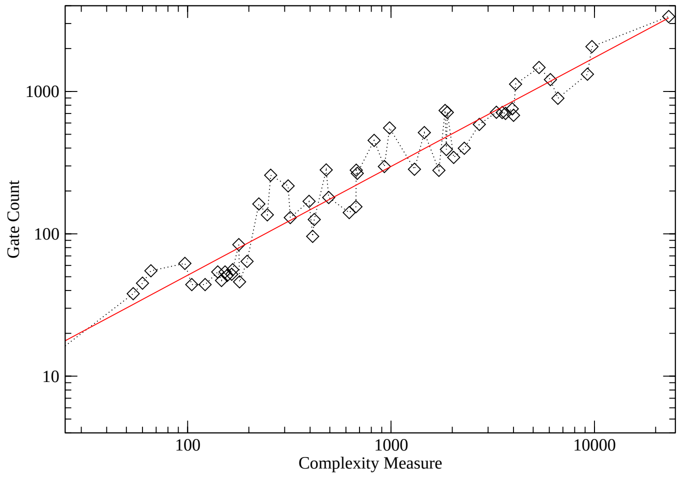

Fig. 3. Gate count - minimum delay point, Synopsys DC, *class* library.

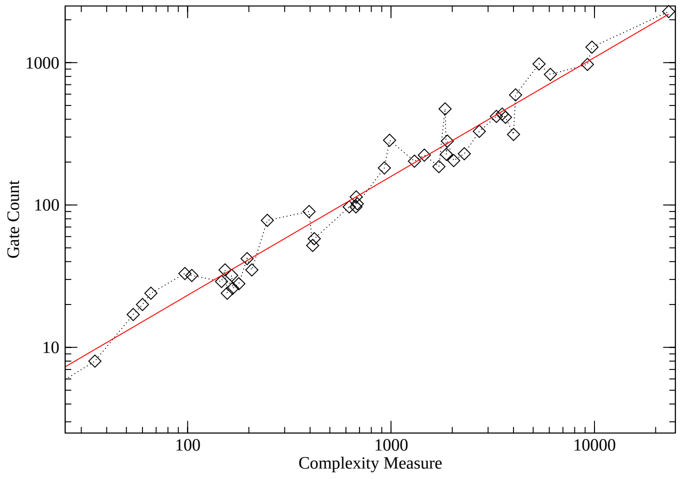

Fig. 4. Gate count - 50% delay point, Synopsys DC, *class* library.

F. Experimental Results

In this section we will report some experimental results, and try to show that our model is general enough to be usable for different design environments.

To show that our model works regardless of the synthesis tool at hand we have generated a plot similar to Figure 2 using SIS [14]. 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*). Figure 5 shows that the relationship between the area complexity measure and the actual area complexity (optimized gate count) can still be approximated by a function of form (7) even though a different synthesis tool is used for optimization/mapping of

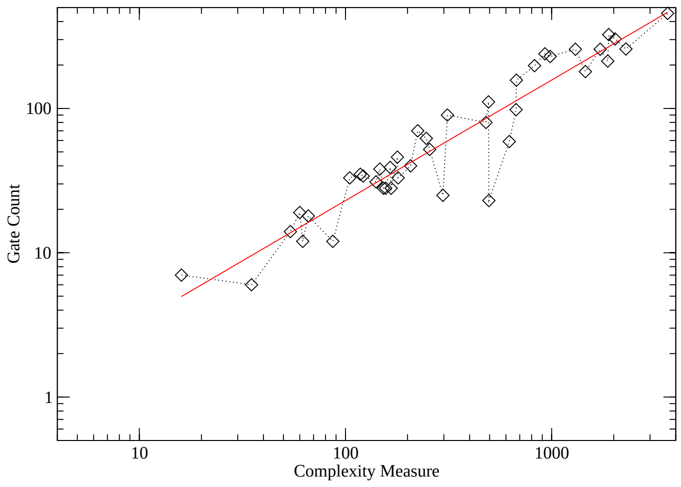

Fig. 5. Gate count - SIS, *class* library.

the benchmarks. The model parameters in this case have of course changed, $m = 0.4911$ and $n = 0.8352$.

Another thing that needs to be checked to verify the usability of our model was whether our model worked for different target gate libraries. To achieve this, we have optimized and mapped the benchmark circuits using Synopsys DC and the Odyssey cell library for the TSMC 0.25μ technology. Once again, we have observed a relationship very similar to what was obtained by earlier experiments, only with different fitting parameters.

To test the accuracy of our model, we 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 Design Compiler.

The set of primitives used for building the BNs were inverters, and 2-input OR gates (the primitive set labeled OR2 in II-C).

For the characterization step, we randomly chose a number of benchmark 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 (4) with these parameters. We have 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 can compute the model parameters m and n, and hence characterize the tool/library/delay point. We can then estimate the optimized gate count for other benchmark circuits using the model obtained by the characterization step and compare these estimates with the optimized gate counts.

Figure 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%.

Figure 7 shows the estimated gate count vs. actual gate count for SIS and the *class* library at the minimum area point. The average error of estimation is 23.3%.

We have also tested the accuracy of our method using the *Odyssey* cell library at the minimum area points as synthesized by Synopsys DC. Once again, we obtained good estimation

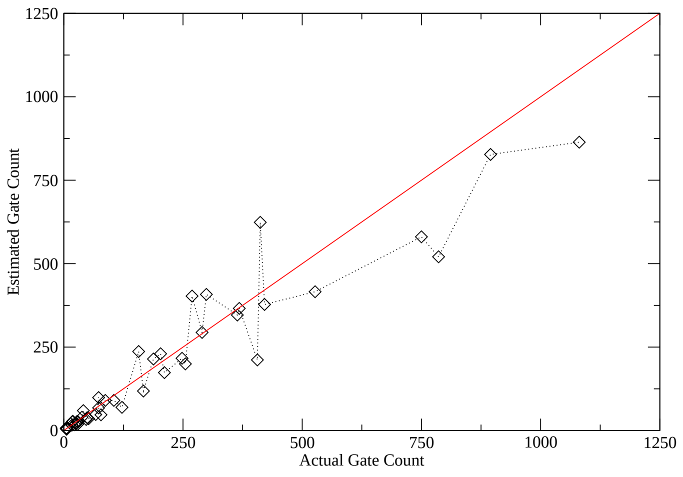

Fig. 6. Gate count - Synopsys DC, *class* library.

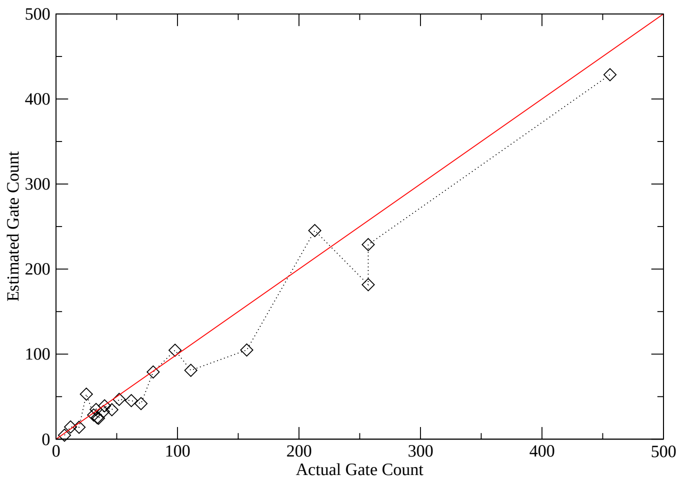

Fig. 7. Gate count - SIS, *class* library.

results with an average error of 25.3%.

G. Limitations of the Model

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.

This work on area estimation can be extended to take the load characteristics into account at the characterization step. The methodology employed to do this would presumably be very similar to the technique employed to characterize the model for different delay constraints.

In the next subsection, we will look at the area model once again, and introduce a tuning technique that will make it more practical.

III. Tuning of the Area Model

In this section, we will introduce a tuning technique for the area model introduced in the previous section. This technique reduces the characterization overhead of the method substantially, and makes it much more practical to use. The results presented here can be found also in [15].

One of the most important features of our high-level complexity model is its tunability. That reduces the characterization overhead of the method substantially, and makes it much more practical to use. Tunability of the model ensures that for any given synthesis tool, we need to perform the computationally expensive characterization step only once (using a simple target library and a simple synthesis script), and use the resulting model for any combination of target libraries and synthesis scripts with only a minimal effort spent on tuning. In this report, we will demonstrate the tuning method and the results obtained by tuning. Our tuning methodology has similarities to the one used in [16] for RTL power models of soft macros.

In the previous section, we introduced a high-level area complexity measure, and a high-level area complexity model based on this measure. The area complexity model is of the form

$$\mathcal{A} = m \cdot C(B)^n \tag{9}$$

where $\mathcal{A}$ is the estimated gate count of the circuit, $C(B)$ is our area complexity measure, m and n are the model parameters that can be obtained by regression analysis. These parameters model the effect of the synthesis tool, the script, target library, and delay specifications on the gate count requirements. This method is very fast once the model parameters m and n are computed for a given *design environment* (to be defined in III-A).

A. Background and Definitions

Definition 3 (technology) *The target gate library that the circuit is mapped on after optimizations.*

Definition 4 (synthesis script) *The settings of the synthesis tools used to optimize the circuit.*

Definition 5 (delay point) *The target synthesis point on the delay/area trade-off curve.*

Definition 6 (design environment) *A specific synthesis script, a specific, and a specific gate library.*

For the rest of the section, we will refer to Synopsys Design Compiler simply as DC, and gate count requirements of the optimized circuit as area.

B. Tuning Methodology

The most computationally expensive step in the area estimation methodology introduced in II, is the up-front characterization of the design environment. Although it is a task which is performed only once for a given design environment, it has to be repeated when a component of the design environment changes. It would be much more practical if it were possible to build the model only once, and somehow "tune" it when the design environment changes, instead of building it from scratch.

In [16], [17], authors show that the relative change in power consumption resulting from a change in technology and/or synthesis script is almost benchmark-independent. As a first step in our tuning approach, we have tried to verify this observation for the gate counts of the benchmark circuits. To do this, we have mapped a set of MCNC benchmark circuits [18] in different design environments. While changing the design environment, we have kept the synthesis tool unchanged (DC). Once we got the area for all the benchmark circuits in different design environments, we tried to find out if there is any correlation between them.

For our experiments on the effect of changes in the target library, we used a large number of different target libraries to map all the benchmark circuits using the default synthesis script of DC at the minimum area point. The result of one such experiment is shown in Figure 8. As can be seen from these plots, the relative change in area caused by a change in technology is indeed almost benchmark independent. That means, we can use a constant scale factor to get the area in one technology based on the area in a different one. This is very promising, since it means we will not have to characterize the changes in design environment resulting from technology changes fully if we can approximate the area model with a linear one.

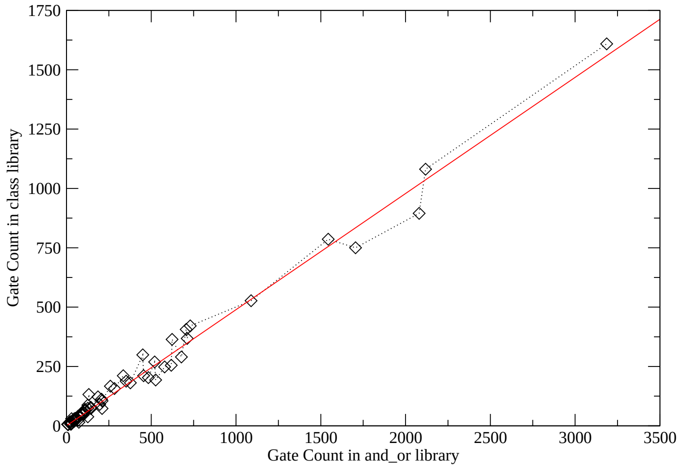

Fig. 8. Gate Count in Class lib. vs Gate Count in and_or lib.

For our second set of experiments, we synthesized the benchmark circuits once again, this time using more aggressive optimization settings (Boolean structuring, and high mapping effort in DC). Then, we investigated the correlation between the area obtained this way and the area obtained using default optimization settings (timing based structuring, medium mapping effort) for different technologies. The result of one of the experiments can be seen in Figure 9.

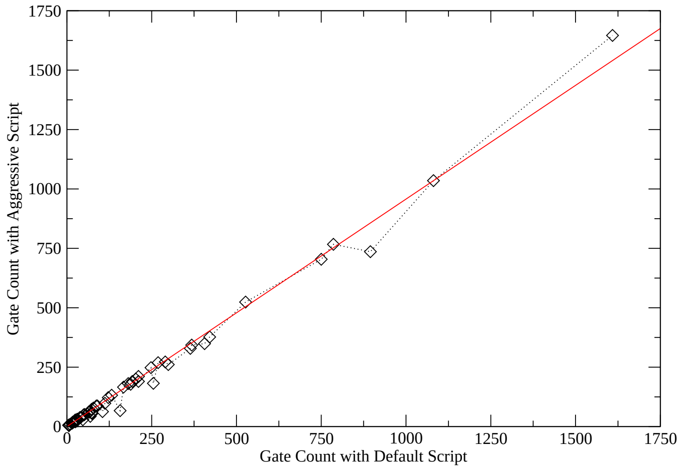

Fig. 9. Gate Count in Class lib. for different synthesis scripts.

Once again, it can be seen that the relative change in area resulting from a change in synthesis script is almost benchmark independent. This result, combined with the result obtained above about technology changes shows that we can handle and change in technology and/or synthesis script by just tuning if we can find a linear area complexity model.

Another thing that can change in the design environment is the delay specifications of the circuit. To test the effect of this on the gate count requirements, we fixed the technology and the optimization settings (except for the delay specifications), and synthesized the benchmark circuits at various delay points. Here, we used the parameterized delay introduced in II-E. The results can be seen in Figure 10. The correlation is almost linear, suggesting that we can tune the model for changes in delay point.

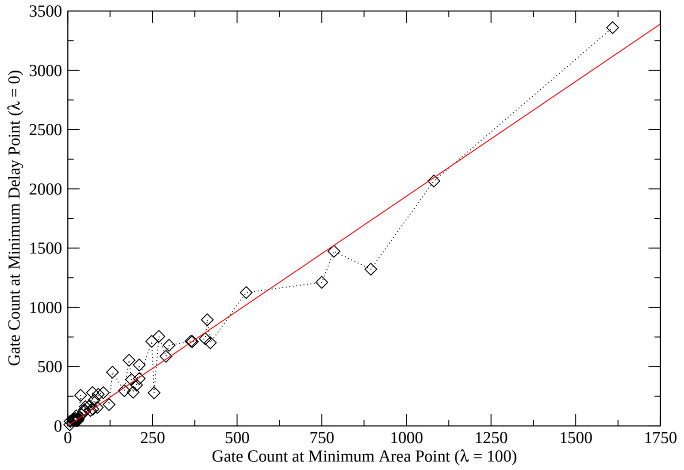

Fig. 10. Gate Count in Class lib for different delay points.

By performing these experiments, we have shown that a linear scale factor can be used to get an estimate for the area of a circuit implemented in a certain design environment based on its area in a different design environment. In our experiments with the model, we have also observed that the parameter n in (9) is very insensitive to the changes in the design environment. This is a very important observation, as it allows us to fix n for a given synthesis tool based on a full characterization step with some design environment, and use a linear model with a modified complexity measure for other design environments using the same synthesis tool. Suppose we have observed that the value of n for a synthesis tool is N_0. Then, we can rewrite (9) as

$$\mathcal{A} = m \cdot C_N(B) \tag{10}$$

where $C_N(B) = C(B)^{N_0}$ is the "modified complexity measure." This model has the advantage of being linear. That means, we can tune this model for any change in design environment by just computing the scale factor of the change and multiplying the parameter m with this factor.

C. Experimental Results

To test our tuning method, we chose a "base" design environment and fully characterized the model (9) for this environment. Then, we used this model and the tuning method described in III-B to estimate the area of the benchmarks in different design environments.

As our "base" environment, we chose the *and_or* library, the default synthesis script of DC, and the minimum area point (*set_max_area* = 0). The model parameters for this particular case was found to be $m = 1.01246$ and $n = 0.809293$.

After the full characterization step, we fixed the value of n to be 0.81, and tuned the model for different environments using scale factors computed in the way described above. One such estimation result can be seen in Figure 11.

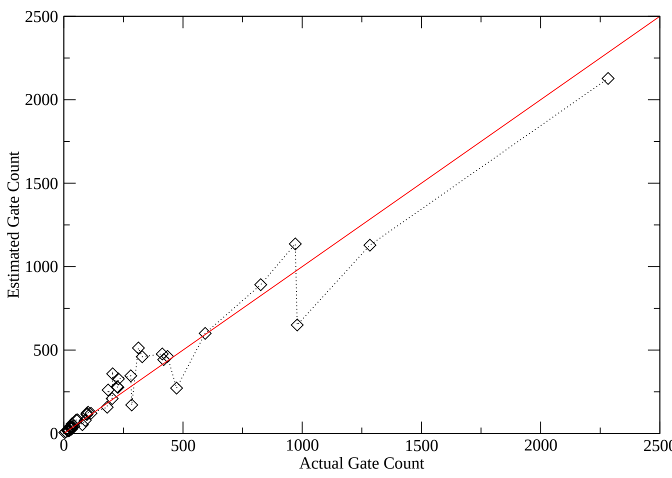

Fig. 11. Gate count for class library - default script at L=50 point.

As can be seen from the figures, the tuning method gives quite acceptable results without going through the expensive characterization phase again and again for every change in the design environment.

IV. Capacitance Estimation

In order to estimate the power, one needs to estimate the average node capacitance in a circuit, $\mathcal{C}_g$, in addition to the area complexity introduced in II and III. In this chapter we present our approach for estimating the average node capacitance. Our approach for estimating the average node capacitance is very similar to the approach presented in [19].

Let $\mathcal{C}_{tot}$ be the total circuit capacitance of an optimal area implementation. Then,

$$\mathcal{C}_g = \frac{\mathcal{C}_{tot}}{\mathcal{A}} \tag{11}$$

where $\mathcal{A}$ is the number of gates in the optimal area implementation. This quantity depends on the target gate library and on the fan-out structure of the circuit. One can try to estimate $\mathcal{C}_g$ by averaging the intrinsic output capacitance of the gates in the target library. This estimate can be made more accurate by weighing the average according to the usage frequency of the gates in a typical design. If one has access to several prior designs which can be used to obtain this data from, the task of estimating the average capacitance becomes much easier. To make the estimate even more accurate, one needs to consider the fan-out structure of the circuit and add the capacitance due to the fan-out branches to the output capacitance of the logic gate. This is the method used in [19], and we use the same method. To do that, we use a number of area optimal circuit implementations in the target library, and obtain the total capacitance. Then, we can use (11) to get the average node capacitance for individual circuits, and perform an average on these numbers to obtain an estimate of $\mathcal{C}_g$ for the library at the minimum area point. The same approach can be employed at different points on the area-delay trade-off curve to obtain the average node capacitance at those points.

V. Activity Estimation

Our high-level activity estimator is based on the estimator introduced in [6]. The work of [6] assumes that the inputs to the RTL block do not have any correlations. In this work, we will propose a modification to this technique that lets it handle correlated input sequences. In the first part of this section, we will summarize the technique of [6], and in the latter parts, we will introduce our modifications.

A. Original Activity Estimator

The problem of estimating the switching activity at the RT level is complicated again by the fact that, there is no gate-level implementation associated with the Boolean function, and hence no notion of "internal nodes" is present. At the RT-level, the best one can do is to run a zero-delay simulation of the Boolean function based-on the user-specified input statistics to obtain the switching activity at the outputs of the combinational block. This, along with very limited structural information, such as the number of inputs and outputs, and user-specified delay constraints, constitutes all the information available to us for deriving an estimate of $\mathcal{D}$.

In [6], [19], authors make two important observations that makes the task of predicting the activity at RT-level possible. First of these observations is about the dependence of average activity on the delay constraints, and the second one is about the quadratic decay of the density at any cross section with increasing depth.

As there are many realizations of a Boolean function with different values of area and delay, it is natural to expect $\mathcal{D}$ to vary across the different implementations of a Boolean function with different delay constraints. However, if one can show that the variation is not significant, this will make the task of deriving an estimator relatively easy, as the implementation details will not have to be taken into account. In [6], authors run a number of experiments that compare the amount of variation in $\mathcal{D}$ across different implementations of the benchmark circuits from the ISCAS 89 [20] and MCNC [18] benchmark suites. Results of this experiments show a maximum absolute deviation of ± 0.05 for an overwhelming majority of the circuits. That means it is possible to use the estimated value of $\mathcal{D}$ as a nominal measure of average switching activity for all implementations of a Boolean function.

A combinational circuit can always be *levelized* so that its gates are tagged with *level* values that represent their distance from the primary inputs. The largest level number K used in levelizing a circuit is called the *circuit depth*. Every circuit node is generated at some unique level and used at possibly several other levels. For every $i = 0, 1, 2, \ldots, K$, define the *set of nodes in cross-section i,* S_i, as the set of all nodes that are generated at levels less than or equal to i and used at levels greater than i.

Definition 7 ($D(i)$**)** *Based on the notion of cross-section, D(i) is defined to be the sum of node densities in the set* S_i*, called the cumulative density at cross-section i or, simply, the density at cross-section i. Thus D(K) is the sum of densities of the primary output nodes denoted by* D_o*. Likewise, D(0) is the sum of densities of the primary input nodes denoted by* D_i*.*

Authors of [6] go on to derive an average activity model Boolean functions:

$$\mathcal{D} \approx \frac{2/3}{n+m}(D_i + 2D_o) \tag{12}$$

where D_i and D_o are the cumulative densities at the input and the output nodes respectively, n is the number of input nodes and m is the number of output nodes.

In spite of all the approximations made in the derivation of (12), it is shown in [19] that the resulting expression works quite well for a broad range of circuits. This expression is essentially the *high-level activity estimator* for [19], [5], and it is also the starting point for our high-level activity model.

B. Limitations of the Original Activity Model

The activity model of V-A works quite well for a broad range of circuits as long as the input vector stream is spatially uncorrelated. In our experiments, we have observed an average estimation error of 25% using this activity model with input streams with low spatial correlation. However, the model become much less accurate as the spatial correlation at the inputs increased, resulting in an estimation error of as much as 40%.

To understand why this is happening, one has to go back to the assumptions of [6], and specifically the assumption that the cumulative density varies quadratically with depth. quadratic dependence of cross-section activity on depth. This assumption is made as a result of gate-level simulations with spatially uncorrelated input streams. We ran a number of experiments to see if this observation holds true in the case of spatially-correlated inputs. As a result of these experiments, we have found out that although the quadratic model approximates the cross-section activity quite well in the case of uncorrelated inputs, it breaks down for highly correlated input streams.

Since spatial correlation is highly probable at the inputs of an RTL block, the high-level activity estimator needs to be improved to take this into account. In the next section, we will propose a technique to modify the activity model of [6] to take the input spatial correlations into account.

VI. Activity Estimation for Correlated Inputs

Before going into the details of our activity estimator, we will refer to [21] for the definitions of *spatial* and *temporal* correlations.

A. Temporal correlation

A signal x is said to be *temporally correlated* if an event (occurrence of certain logic state) at a given time is correlated to an event at some past time. In this work, we will concentrate only on correlations across one clock edge. For temporally correlated primary inputs, temporal correlation parameter for the ith input, TC_i, is defined as

$$TC_i = \mathcal{P}\left\{x_i^t \wedge x_i^{t-1} = 1\right\} \tag{13}$$

where $t-1$ and t are consecutive clock cycles and where $\mathcal{P}\{\cdot\}$ denotes probability. Temporal correlation coefficient (γ_i) for ith input is defined as [22]

$$\gamma_i = \frac{\mathcal{P}\left\{x_i^t \wedge x_i^{t-1} = 1\right\} - P(x_i)^2}{P(x_i)(1 - P(x_i))} \tag{14}$$

where $P(x_i)$ is the probability at an input node x_i, and the only quantity which is unknown in (14) is $\mathcal{P}\left\{x_i^t \wedge x_i^{t-1} = 1\right\}$. Therefore it is possible to estimate γ_i if TC_i can be determined. In [21], authors show that TC_i can actually be determined from the knowledge of $P(x_i)$ and $D(x_i)$, and hence temporal correlation at the primary inputs is taken care by $P(x_i)$ and $D(x_i)$ without a need to introduce an additional parameter to represent it. The relationship between TC_i, $P(x_i)$ and $D(x_i)$ is given by

$$TC_i = P(x_i) - \frac{D(x_i)}{2} \tag{15}$$

B. Spatial correlation

A signal x is said to be *spatially correlated* to another signal y if their events are correlated. In this work, we will concentrate only on pairwise correlations. Once again, referring to [21], we can define SC_{ij}, the spatial correlation between the ith and jth inputs as

$$SC_{ij} = \mathcal{P}\left\{x_i \wedge x_j = 1\right\} \tag{16}$$

i.e., the probability of the inputs being high simultaneously.

This definition of SC_{ij} as a measure of spatial correlation follows from the definition of the correlation coefficient as introduced in [22]

$$\rho_{ij} = \frac{\mathcal{P}\{x_i \wedge x_j = 1\} - P(x_i)P(x_j)}{\sqrt{P(x_i)P(x_j)(1-P(x_i))(1-P(x_j))}} \tag{17}$$

From the definition given in (16), it is clear that SC_{ij} is sufficient to capture ρ_{ij}.

Instead of considering all the pairwise correlation coefficients, it is possible to define SC_{in} (*average spatial correlation coefficient*, i.e., average of all SC_{ij} terms). This parameter can be calculated as

$$SC_{in} = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathcal{P}\{x_i = 1, x_j = 1\} \tag{18}$$

where n is the number of primary inputs.

In [21], authors go on to find upper and lower bounds for SC_{in} as

$$\frac{nP_{in}^2 - P_{in}}{(n-1)} \le SC_{in} \le P_{in} \tag{19}$$

where P_{in} is the average signal probability for primary inputs.

For our work, we will use a parameterized measure of spatial correlation instead of directly using SC_{in}.

Definition 8 (parameterized spatial-correlation coefficient, λ_{sc}) *For a given P_{in}, let SC_{min} be the lower bound of the average spatial correlation coefficient SC_{in} and SC_{max} be the upper bound. The spatial correlation parameter corresponding to an arbitrary spatial correlation value SC can be computed as*

$$\lambda_{sc} = \frac{(SC - SC_{min})}{(SC_{max} - SC_{min})} \tag{20}$$

We observed that the parameter, λ_{sc}, captures the effect of spatial correlation on switching activity closely.

C. Modified activity model

In V-A, we introduced a high-level activity model. During our experiments with correlated inputs, we observed that this model can be modified to capture the switching activity with input vector streams with spatial correlation. Our proposed *high-level activity estimator* is

$$\mathcal{D}_{sc} = \mathcal{D}/S(\lambda_{sc}, P_{out}^{*}, \mathcal{A}) \tag{21}$$

where $\mathcal{D} = (2/3)(D_i + 2D_o)/(n+m)$ is the activity estimated by the original model, $S(\cdot)$ is the "correction factor", λ_{sc} is the parameterized spatial-correlation coefficient, P_{out}^{*} is the average signal probability of the primary outputs for completely random inputs, and $\mathcal{A}$ is the gate count of the minimum area implementation of the Boolean function (or an estimate of it).

To determine the form of $S(\cdot)$, we ran a number of experiments on a number of MCNC [18] benchmark circuits. We generated a very large number of input vector streams with varying statistics, and ran zero-delay gate-level simulations on minimum area gate-level implementations of these circuits to obtain simulated average activity numbers. Then, we used (12) to estimate the average activity for the same circuits, and simply divided the activity numbers obtained from the simulation by the estimated activity numbers to obtain the correction factor. Figure 12 show the results of one such experiment.

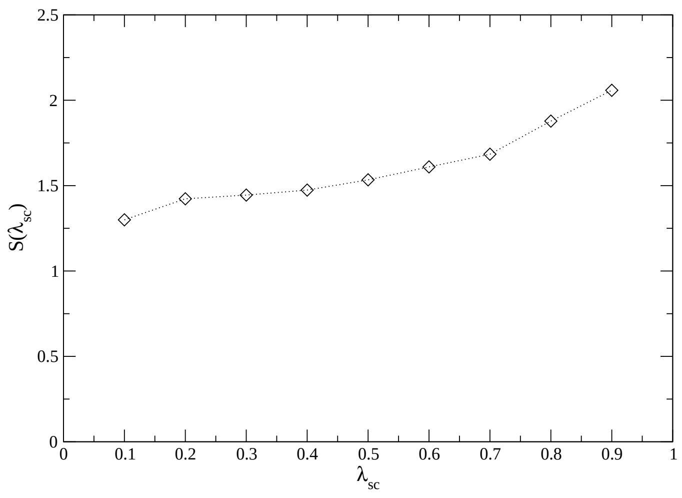

Fig. 12. Correction factor for C1355.

During these experiments, we have discovered that the correction factor, $S(\cdot)$ is not a constant factor across different circuits.

D. Determining the form of $S(\cdot)$

There are two alternatives for studying the variation in the form of $S(\cdot)$ across different circuits. One can use actual benchmark circuits to run the experiments explained in VI-C to study the variations, and try to come up with a technique. We have found this approach to be of limited use, as there are only a limited number of standard benchmark circuits to work on. The second alternative approach, which is also our choice, is to use *randomly generated Boolean functions* [19] to study the variations, and then find a mapping between RGBFs and real functions.

Definition 9 (randomly generated Boolean function (RGBF)) *An RGBF is a Boolean function selected by randomly assigning each point in the Boolean space to either the on-set or off-set of the function.*

The advantage of using RGBFs for studying (and characterizing) the form of $S(\cdot)$ is obvious: one can generate a very large number of different RGBFs to work on. However, to be of practical use, the characterization performed on these functions should be applicable to real Boolean functions. In [19], [5], authors show that RGBFs and real Boolean functions behave similarly in terms of area complexity. We will show later in this chapter that the same is true for the form of the "correction factor" for the high-level activity model.

The first step in analyzing the variations in $S(\cdot)$ is to generate a large number of RGBFs. These RGBFs would have number of primary inputs in a certain range (depending on the sizes of the target circuits), and output probabilities (to be defined later) that span the entire range from 0 to 1. The second step, then, is to synthesize these functions (we used Synopsys Design Compiler for this task) to obtain a minimum area gate-level implementation. Once the circuits are available, one can run a large number of experiments similar to the one introduced in VI-C to generate the correction factors. Figure 13 shows the results of experiments run on RGBFs of input size 11.

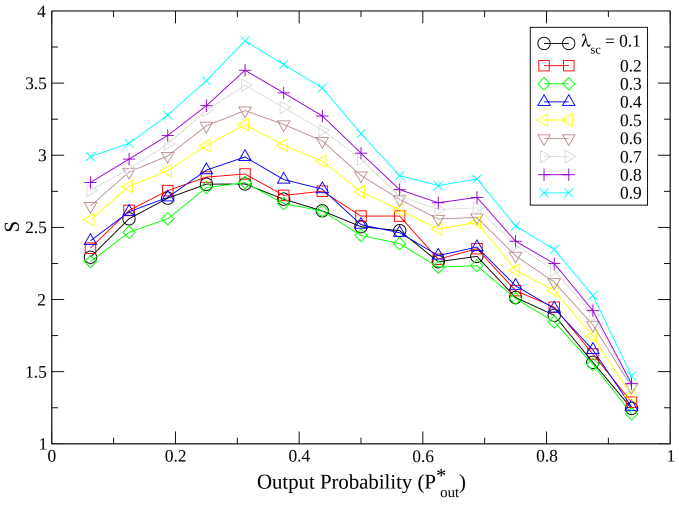

Fig. 13. Correction factor for RGBF with n=11.

It is obvious from the results presented that the correction factor $S(\cdot)$ is a function of spatial correlation (λ_{sc}), output probability and the input support size of the RGBF (which, combined with the output probability determines the area of the gate-level implementation).

Definition 10 (output probability, P^*_{out}) *The average signal probability at the primary outputs of a Boolean function for a completely random ($P_{in} = 0.5$, $D_{in} = 0.5$) input stream.*

To capture the values of the correction factor, we employed a look-up table (LUT) based approach. The look-up table proposed is a three-dimensional look-up table indexed by P^*_{out}, $\mathcal{A}$ and λ_{sc}. The reason $\mathcal{A}$ is used instead of n (number of primary inputs) as an index to the table is that, $\mathcal{A}$ covers a larger parameter space than n. If we were to choose n as an index, we would be forced to generate data for the entire range of n (which can be very large). In our case, by generating data for only $6 \leq n \leq 13$ we were able to span a range of $8 \leq \mathcal{A} \leq 1594$.

E. Table look-up algorithm

Obviously, one cannot include every combination of $P^*_{out} - \mathcal{A}$ values in the look-up table. Therefore, one needs a table look-up algorithm which performs interpolation of data points as needed. We have found an algorithm that works with reasonable accuracy to be:

- Obtain P^*_{out} for the Boolean function at hand
- Choose the column indexed by the numerically closest P^*_{out} value in the LUT
- Choose the two closest $\mathcal{A}$ points in this column to the gate count estimate of the Boolean function
- Use linear interpolation on the values of S according to the relative position of the circuit $\mathcal{A}$ between these two data points

F. Polynomial approximation to the LUT

In VI-D, we mentioned that we chose to employ a look-up table based approach for modeling the correction factor. Obviously, one may choose to fit a curve to the data in the table instead of using the table directly. This way, the need for the table look-up algorithm introduced in the previous section can be eliminated and the computation of the correction factor can be made much faster and with less memory consumption.

To test this approach, we chose to employ polynomial regression due to its simplicity. Our regression analysis showed that a third degree polynomial on three variables ($\mathcal{A}$, P^*_{out}, and λ_{sc}) models the look-up table very closely.

G. Experimental Results

In this section, we will present the results of the experiments we have performed to verify our *high-level activity estimator*.

1) Verification of the LUT approach: To verify the validity of our LUT approach, we ran a number of experiments on MCNC benchmark circuits to see if the values in the LUT indexed by the P^*_{out}, $\mathcal{A}$ and λ_{sc} matched the values of the correction factor obtained by gate-level simulation. Figure 14 show the results of one such experiment. As can be seen, there is a good agreement between the LUT values and the simulation results.

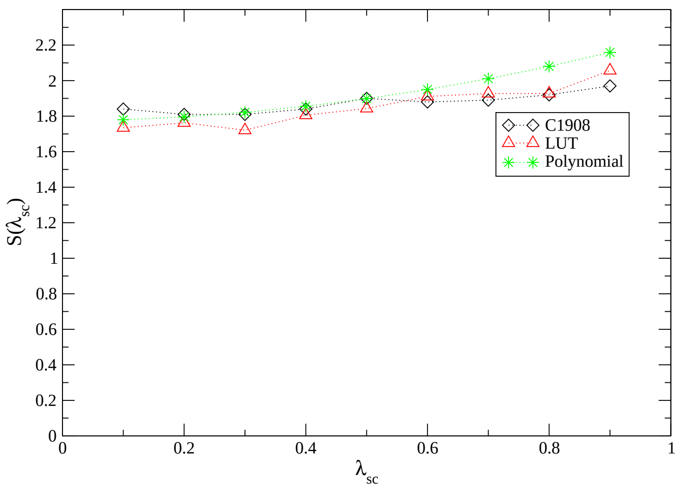

Fig. 14. Estimated correction factors for C1908.

2) Verification of the Polynomial Approximation: To test the accuracy of the polynomial approximation to the look-up table we ran experiments on the MCNC benchmark circuits to see how well the polynomial approximates the actual correction factor. Figure 14 shows the result of one such experiment. As seen from the plot, the polynomial correction factor models the actual correction factor very closely.

3) Verification of the activity model: Finally, to verify that our high-level activity estimator works in the presence of spatial correlation at the inputs, we ran experiments to compare the simulated and estimated activity values for a number of benchmark circuits. To be able to determine the improvement of estimation accuracy with respect to the original estimation method, we have repeated these experiments once with the original estimation method, and once with the modified activity model introduced in this work. Figure 15 is a correlation plot for estimation using the modified activity model. The average error of estimation is 21.6% for this experiment. For comparison, the average error of estimation obtained using the original activity method for the same input statistics is 32.7%.

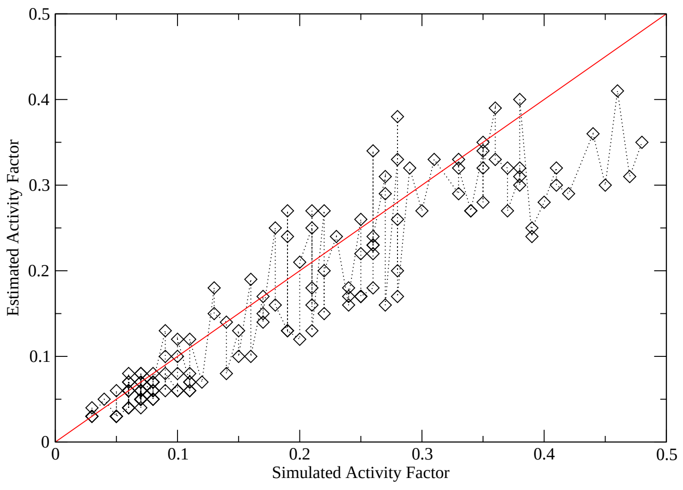

Fig. 15. Activity estimation with new method and correlated inputs.

VII. High-Level Power Estimation

In the previous sections we addressed the problem of estimating the area complexity and the average activity of Boolean functions from a high-level view. We can now combine the estimated average activity with an estimate of the area (capacitance) using (2) to obtain an estimate of the power dissipated by the Boolean function. In this chapter, we will outline the full power estimation flow introduced so far and present the results obtained using this flow. A statistical power estimation tool based on [23] was used to measure the power dissipated by the gate-level designs. This power estimator is basically a Monte Carlo based node transition density estimator. The tool generates and simulates vector streams with user defined statistics while observing the individual node transition densities and stops once all the estimated node densities are within the user specified accuracy and confidence levels. That means, the tool will generate vector streams of different lengths for each circuit at hand which are as long as necessary to achieve the user defined accuracy levels.

For our experiments, we ran the gate level power estimator for 5% error tolerance at 95% confidence level. For each circuit, we have simulated the gate level circuits for a wide range of input statistics, varying signal probability, activity factor and pair-wise spatial correlation values.

A. *Combinational Power Estimation Flow*

The proposed power estimation flow for combinational logic blocks based on the results presented so far and (2) can be summarized as follows:

1) Read in the Boolean equations describing the design
2) Read in the input statistics and delay constraints
3) Build a Boolean network representation of the design
4) Estimate the gate count using the technique introduced in II
5) Estimate the total capacitance using the technique introduced in IV
6) Run a zero-delay logic simulation of the design with random inputs to obtain P^*_{out}
7) Estimate the average activity of the circuit using the technique of V
8) Estimate the power using (2)

B. *Results: Power Estimation*

We begin by comparing the power estimated using the original activity estimator with power dissipated by a gate-level minimum-area implementation under zero-delay conditions. The results of this comparison are shown in Figure 16. For this experiment, we have used input streams with various statistics. Among these streams, there are ones with high spatial correlations, as well as low spatial correlations.

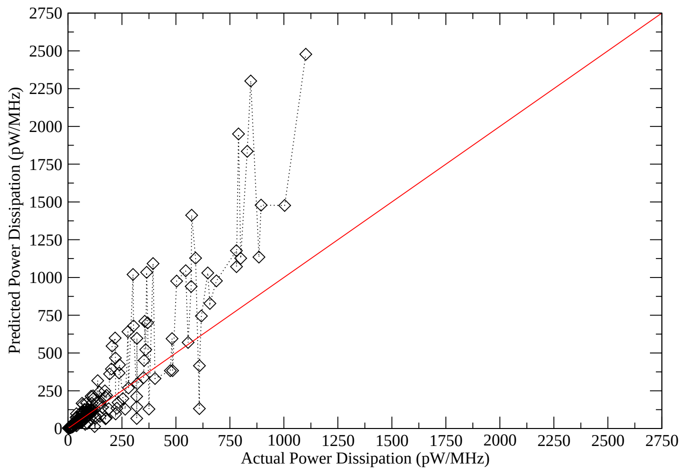

Fig. 16. Actual and predicted zero-delay power obtained using the original activity model.

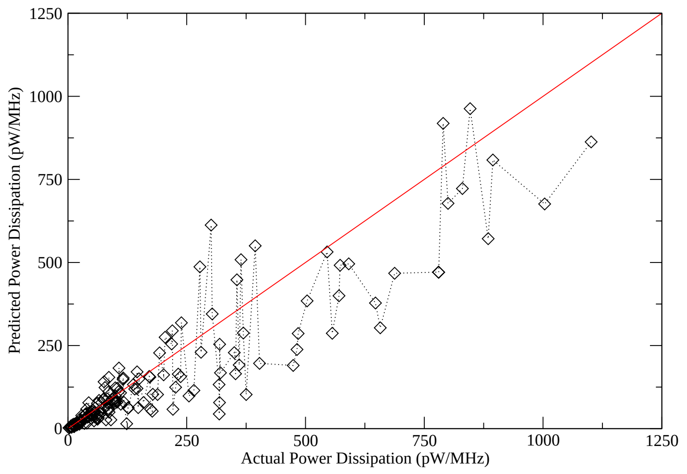

Fig. 17. Actual and predicted zero-delay power obtained using the modified activity model.

The average error of prediction with this method is 46.9%. As can be seen from the plot, this technique has a tendency to overestimate the power dissipation.

In Figure 17, we are presenting the results of the same experiment using the modified activity model introduced in V. The average prediction error is 32.5% for this experiment.

VIII. Conclusion

In this paper, we have presented the components of a high-level power estimation technique. The first component presented was a new area estimation technique based on the complexity of the Boolean network representation of the high-level design. Then, we have proposed a methodology for estimating $\mathcal{C}_g$ needed to convert the area estimate into an estimate of the total capacitance. This, combined with the proposed activity estimator can be used to get high-level power estimates at the structural RT level.

References

[1] P. Patra, "A perspective on power issues in Ultra DSM circuits," Invited talk presented at Int'l Conference on Information Technology, CIT 2000, 2000.

[2] S. Powell and P. Chau, "A model for estimating power dissipation in a class of DSP VLSI chips," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 6, pp. 646–650, 1991.

[3] P. E. Landman and J. M. Rabaey, "Architectural power analysis: The dual bit type method," *IEEE Transactions on VLSI Systems*, vol. 3, no. 2, pp. 173–187, June 1995.

[4] S. Gupta and F. N. Najm, "Analytical models for RTL power estimation of combinational and sequential circuits," *IEEE Transactions on Computer-Aided Design*, vol. 19, no. 7, pp. 808–814, July 2000. [Online]. Available: http://www.eecg.toronto.edu/ najm/papers/tcad00-gupta.pdf

[5] M. Nemani and F. N. Najm, "High-level area and power estimation for VLSI circuits," *IEEE Transactions on Computer-Aided Design*, vol. 18, no. 6, pp. 697–713, June 1999. [Online]. Available: http://www.eecg.toronto.edu/ najm/papers/tcad99-mahadev.pdf

[6] ——, "Towards a high-level power estimation capability," *IEEE Transactions on Computer-Aided Design*, vol. 15, no. 6, pp. 588–598, June 1996. [Online]. Available: http://www.eecg.toronto.edu/ najm/papers/tcad96-entropy.pdf

[7] F. N. Najm, "Transition density: A new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design*, vol. 12, no. 2, pp. 310–323, Feb. 1993. [Online]. Available: http://www.eecg.toronto.edu/ najm/papers/tcad93-densim.pdf

[8] K. M. Buyuksahin and F. N. Najm, "High-level area estimation," in *Proc. International Symposium on Low Power Electronics and Design (ISLPED)*, Monterey, CA, Aug. 2002, pp. 271–274. [Online]. Available: http://www.eecg.toronto.edu/ najm/papers/islped02-kavel.pdf

[9] K. A. Bartlett, R. K. Brayton, G. D. Hachtel, R. M. Jacoby, C. R. Morrison, R. L. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang, "Multilevel logic minimization using implicit don't cares," *IEEE Transactions on Computer-Aided Design*, vol. 7, no. 6, pp. 723–740, June 1988.

[10] C. E. Shannon, "The synthesis of two-terminal switching circuits," *Bell System Technical Journal*, vol. 28, no. 1, pp. 59–98, 1949.

[11] D. E. Muller, "Complexity in electronic switching circuits," *IRE Trans. on Electronic Computers*, vol. 5, no. 1, pp. 15–19, 1956.

[12] N. Pippenger, "Information theory and the complexity of Boolean functions," *Mathematical Systems Theory*, vol. 10, no. 1, pp. 129–167, 1977.

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

[14] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Electronics Research Laboratory, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720, Memorandum UCB/ERL M92/41, May 1992.

[15] K. M. Buyuksahin and F. N. Najm, "On the tunability of a high-level area model," Coordinated Science Laboratory - U of Illinois at Urbana-Champaign, 1308 West Main Street, Urbana, IL 61801, Tech. Rep. UILU-ENG-02-2225 DAC 95, Oct. 2002.

[16] A. Bogliolo, R. Corgnati, E. Macii, and M. Poncino, "Parametrized RTL power models for soft macros," *IEEE Transactions on VLSI Systems*, vol. 9, no. 6, pp. 880–887, Dec. 2001.

[17] A. Bogliolo and L. Benini, "Node sampling: a robust RTL power modeling approach," in *Proc. International Conf. Computer-Aided Design (ICCAD)*, 1998, pp. 461–467.

[18] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide - Version 3.0*, Microelectronics Center of North Carolina (MCNC), P. O. Box 12889, Reseach Triangle Park, NC 27709, Jan. 1995.

[19] M. Nemani, "High-level power estimation," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Feb. 1998.

[20] D. B. F. Brglez and K. Koźmiński, "Combinational profiles of sequential benchmark circuits," in *Proc. International Symposium on Circuits and Systems*, 1989, pp. 1929–1934.

[21] S. Gupta and F. N. Najm, "Power modeling for high level power estimation," *IEEE Transactions on VLSI Systems*, vol. 8, no. 1, pp. 18–29, Feb. 2000. [Online]. Available: http://www.eecg.toronto.edu/ najm/papers/tvlsi00-gupta.pdf

[22] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd ed. New York: McGraw-Hill, 1991.

[23] M. G. Xakellis and F. N. Najm, "Statistical extimation of the switching activity in digital circuits," in *Proc. ACM/IEEE Design Automation Conference*, San Diego, CA, 1994, pp. 728–733. [Online]. Available: http://www.eecg.toronto.edu/ najm/papers/dac94-med.pdf

PLACE PHOTO HERE

Kavel M. Büyükşahin received the B.S. degree in Electrical and Electronic Engineering from Bogazici University at Istanbul, Turkey in 1998, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering (ECE) from the University of Illinois at Urbana-Champaign (UIUC) in 2000 and 2003, respectively. Since 2003, he has been at Intel Corporation in Hillsboro, OR, as a Senior CAD Engineer with the Technology CAD Department.

Dr. Büyükşahin is a Member of the IEEE. His research interests include CAD tool development for power estimation and modeling, signal integrity analysis, and performance verification.

PLACE PHOTO HERE

Farid N. Najm received the B.E. degree in Electrical Engineering from the American University of Beirut (AUB) in 1983, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering (ECE) from the University of Illinois at Urbana-Champaign (UIUC) in 1986 and 1989, respectively. He worked as Electronics Engineer with AUB from 1983 to 1984. In 1989, he joined Texas Instruments in Dallas, TX, as Member of Technical Staff with the Semiconductor Process and Design Center. In 1992, he joined the ECE Department at UIUC as an Assistant Professor, and then became a Tenured Associate Professor at UIUC in 1997. In 1999, he joined the ECE Department at the University of Toronto, where he is currently Professor and Vice-Chair of ECE.

Dr. Najm is a Fellow of the IEEE, and is Associate Editor for the IEEE Transactions on CAD. He received the IEEE Transactions on CAD Best Paper Award in 1992, the NSF Research Initiation Award in 1993, the NSF CAREER Award in 1996, and was Associate Editor for the IEEE Transactions on VLSI 1997–2002. He served as General Chairman for the 1999 International Symposium on Low-Power Electronics and Design (ISLPED-99), and as Technical Program Co-Chairman for ISLPED-98. He has also served on the technical committees of ICCAD, DAC, CICC, ISQED, and ISLPED. Dr. Najm has co-authored the text "Failure Mechanisms in Semiconductor Devices," 2nd Ed., John Wiley & Sons, 1997. His research interests are in the general area of CAD tool development for low-power and reliable VLSI circuits, including power estimation and modeling, low-power design, power grid analysis and verification, and reliability analysis and prediction.