Generating Synthetic Benchmark Circuits for Evaluating CAD Tools

Dirk Stroobandt, *Member, IEEE*, Peter Verplaetse, *Student Member, IEEE*, and Jan Van Campenhout, *Member, IEEE*

***Abstract*—For the development and evaluation of computer-aided design tools for partitioning, floorplanning, placement, and routing of digital circuits, a huge amount of benchmark circuits with suitable characteristic parameters is required. Observing the lack of industrial benchmark circuits available for use in evaluation tools, one could consider to actually generate synthetic circuits. In this paper, we extend a graph-based benchmark generation method to include functional information. The use of a user-specified component library, together with the restriction that no combinational loops are introduced, now broadens the scope to timing-driven and logic optimizer applications.**

Experiments show that the resemblance between the characteristic Rent curve and the net degree distribution of real versus synthetic benchmark circuits is hardly influenced by the suggested extensions and that the resulting circuits are more realistic than before. An indirect validation verifies that existing partitioning programs have comparable behavior for both real and synthetic circuits. The problems of accounting for timing-aware characteristics in synthetic benchmarks are addressed in detail and suggestions for extensions are included.

***Index Terms*—Net degree distribution, Rent's rule, synthetic benchmark generation, timing-driven applications.**

I. Introduction

THE PRODUCTION of very large scale integrated (VLSI) chips requires the layout (floorplanning, placement and routing) of the chip design on a carrier. With the advent of high level description languages such as VHDL, with the extensive use of component libraries, and with the standardization of production parameters, more and more steps in the design cycle are being automated. Computer-aided design (CAD) tools have become indispensable to cope with the complexity and the limited time resources.

For the high demands put on system performances these days, CAD tools often lack flexibility. Improving the existing CAD tools, therefore, remains necessary. New algorithms for (timing-driven) partitioning, floorplanning, placement, routing, etc. (we refer to such applications as "*partitioning applications*") should be evaluated thoroughly and this entails the need for "good" evaluation tools. Crucial to this evaluation is the use of a very large set of benchmark circuits that consists of a sample of the circuits at which the CAD tool is aimed.

Initiatives for distributing benchmark circuits have been taken [1], [2]. However, most sets of benchmark circuits used in the research community today are fairly small. Moreover, these benchmark circuits are often not large enough to be useful for the complex tools we want to evaluate today. Last but not least, they often do not have the right parameter characteristics. For instance, the ISCAS85 benchmark circuits were intended specifically for evaluating automatic test pattern generation (ATPG) tools and, hence, contain special structures that might deviate from what can be expected in a general circuit. Benchmark sets must ensure that the benchmarks they contain differ enough to span a broad range of all possible circuits a tool can expect during its lifespan. The first requirement for this is to be able to separate out the most important circuit characteristics (for a certain tool) and to measure those characteristics from the benchmarks. The next requirement, and the most problematic one, is to find enough example circuits that cover a broad range of each of those characteristics. This proves to be very hard with real-life benchmark circuits because one cannot control their characteristics. For this reason, benchmark sets even have to be larger than the minimum possible set to cover the entire range of properties sought after. Because of the proprietary nature of industrial circuits, it is almost impossible to compile sufficiently large sets of realistic circuits.

New sets of benchmark circuits are definitely needed for evaluating new tools that are developed in several research groups. Only recently, the generation of synthetic benchmark circuits is becoming to be recognized as a viable alternative. Their main asset is that they provide full control over some of the benchmark's important characteristic parameters, such as circuit size, interconnection structure and functionality. Ideally, these parameters can be set independently, and one has full control over the granularity of the synthetic benchmark suites. For example, one can generate a set of circuits with increasing circuit size while maintaining the pin count. The major drawback of synthetic benchmark suites is that it is hard to prove that a set of circuits is representative for all or at least a class of circuits for a given application, since usually not all aspects are modeled in a realistic way. The validation process is usually omitted when a benchmark generation method is presented in literature. A general framework for such a validation is presented in [3] and we use some of its ideas to validate our results in this paper.

In [4], Stroobandt *et al.* suggested that two parameters are fundamental for obtaining viable circuits: the Rent exponent (as a measure of the interconnection complexity) and the net degree distribution. They presented a method of generating synthetic circuits with similar Rent and net degree parameters as for real circuits. In this paper, we seek to include some basic timing properties in the synthetic benchmarks, since timing-driven partitioning and placement is a major concern these days. Whereas previous efforts for benchmark generation mainly focused on

Manuscript received November 27, 1999. This paper was recommended by Associate Editor M. Wong.

D. Stroobandt and P. Verplaetse are with the Fund for Scientific Research (F.W.O.), Flanders, Belgium. They are also with Ghent University, B-9000 Ghent, Belgium.

J. Van Campenhout with Ghent University, B-9000 Ghent, Belgium.

Publisher Item Identifier S 0278-0070(00)07473-X.

graph-based properties of circuits, we wish to combine the best of graph-based properties with the possibility to include timing information. Basically, what is missing in the graph-based view, is *functionality.* Logic optimizer tools and test generation programs need information on the function of the gates, placement tools need information on the block area, and routers need information on the terminal positions. It is, therefore, desirable to know the specific type of the individual gates. This is the basic information the circuit has to contain for timing-driven tools to be able to run on them properly. Of course, introducing functionality requires a correct behavior at the logic level, hence the generated circuits should be free from combinational loops. But this is only part of the picture; even more is required for benchmarks to be useful for real timing-driven applications. One example is control over the path length distributions, which will be the topic of further research in this area. In this paper, we acknowledge the need for timing-aware benchmarks and we evaluate the problems created by adding functionality to graph-based synthetic benchmark generation methods. We also show that by introducing functionality carefully, the synthetic benchmarks can be made more realistic, even for nontiming-driven applications.

To obtain this goal, we extend our previous benchmark generation program `gnl`[1] [4] by enabling the choice of gates from a user-defined library. The inclusion of flip-flops or latches also enables us to generate sequential circuits. Special care is taken to exclude connections that introduce combinational loops and the effects of this are studied. The experimental results show that the exclusion of combinational loops does not fundamentally change the characteristic parameters, such as the Rent curve and the net degree distribution. An exception is to be made for the number of primary inputs and primary outputs that can deviate under extreme circumstances. A number of experiments investigates when this occurs and what is the underlying reason for it.

The direct validation of the Rent curve and the net degree distribution shows that our generated benchmark circuits are viable alternatives for real circuits. Equally important is an indirect validation of the synthetic benchmarks. Since the benchmarks are intended to be used for evaluating CAD algorithms, the indirect validation can be done by running several algorithms on both an original benchmark and its synthetic counterpart and comparing the results. Our experiments verify the usefulness of our synthetic benchmarks for what we call partitioning applications. This paper presents a way to extend the application domain of our benchmarks to timing-aware tools, discusses some fundamental problems inherent to this extension and addresses some issues as to how one can mitigate these problems.

In Section II, we discuss related work on benchmark generation. Section III gives an overview of the parameters that characterize real circuits and Section IV describes the basic procedure of generating benchmark circuits. The extension of our benchmark generation method toward timing-aware characteristic parameters is the subject of Section V. Section VI is devoted to interesting experimental results and in Section VII the problems related to including timing characteristics in the benchmarks are addressed. Possibilities for future extensions to mitigate these problems are suggested.

[1] `gnl` is the acronym for "**g**enerate **n**et**l**ist"

II. Related Work

Mainly because of the lack of available industrial benchmarks, the research community has tried to come up with different ways of generating synthetic circuits (some of them are presented in [5, p. 81]). An obvious way is to select a number of logic gates and then connect the gate terminals randomly with a certain probability. Unfortunately, it is questionable if such "circuits" can be accepted as realistic logic circuits. For this reason, the first successful trials of benchmark generation were not based on making random connections between gates but on applying a sequence of random transformations on an initial (existing) circuit [6]. The advantage over complete random circuits was that the resulting "circuits" were more realistic. However, the problem then is that virtually no attributes of the circuits can be controlled, again leading to not really viable circuits for the intended application domain. In a succeeding paper [7], Iwama *et al.* tried to alleviate the problem by at least setting bounds to the gate fan-ins. Ghosh *et al.* [8] suggested a similar benchmark generation technique by searching for "mutants" of existing circuits with the same "signature." They restrict their method to changing interconnections between different circuit levels, retaining the number of gates per level. This results in Wiring signature-invariant mutants. In [9], their method has been augmented with Entropy Signature-Invariant mutations.

Hutton *et al.* [10] addressed the problem of random generation of combinational circuits "from scratch." They defined properties such as size, delay, physical shape, edge-length distribution, and fanout distribution to describe the physical characteristics of a purely combinational circuit and generated circuits with an exact parameterization ("clones" of existing circuits). By comparing characteristics of the generated circuit that are not specified as parameters to generation (reconvergence and routability), they showed that the generated circuits behave very comparably to real circuits, whereas random circuits of the same size do not. In a consecutive paper [11], they further extended their method to include sequential circuits.

In [12], Pistorius *et al.* noticed that the previously described generation methods were not able to fully match the (hierarchical) structure of real designs. They presented a method of generating benchmarks that uses the hierarchical structure explicitly and their results show that, contrary to the other methods, their synthetic benchmark circuits (aimed at field programmable gate arrays) have similar filling rates as real circuits.

For the evaluation of CAD tools, only a finite set of particular benchmark circuits can be used. There is no way of proving that algorithms performing well for this set are suitable for every circuit. This would require an immensely huge set of benchmark circuits. Therefore, new algorithms can only be tested efficiently on their merits by a careful evaluation with respect to those characteristic parameters of circuits that are felt to be most important for the particular application [3]. This allows the use of a smaller set of benchmark circuits provided they have characteristics "on demand." The main advantage of synthetic benchmark circuits is the controllability of a single characteristic parameter at a time, with limited influence on the other parameters. This feature enables us to draw much more funded conclusions from experimental results. The main problem remains to decide what are the characteristic parameters of circuits that have to be controlled in order to obtain viable benchmark circuits.

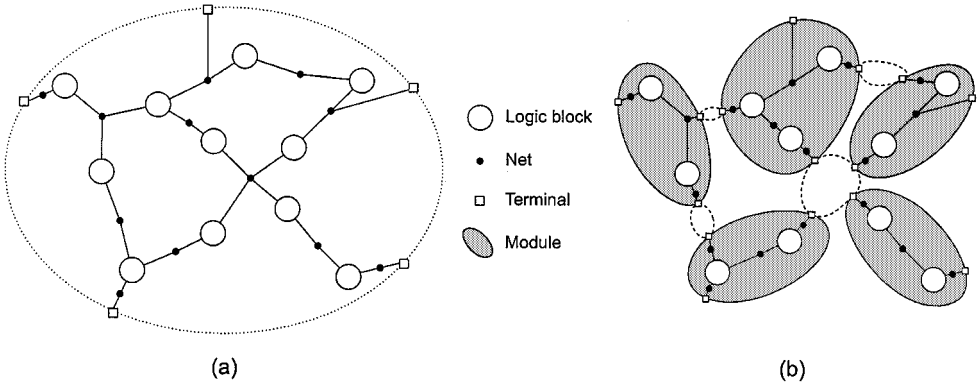

Fig. 1. Model of a circuit (a) and the partitioning of the circuit into modules (b).

For the evaluation of algorithms that are related to partitioning, the interconnection complexity is the main characterization parameter [4], [13] that captures the structural information for interconnections. It is reflected through Rent's rule [14], which is a relationship between the number of terminals in a partitioned circuit and the number of blocks per partition (see Section III). Real circuits have been shown to follow Rent's rule, so controlling the Rent exponent of synthetic benchmark circuits is a necessity. Darnauer and Dai [15] were the first to attempt to generate random benchmark circuits, based on Rent's rule. Their program, called `rmc`,[2] generates large random circuits with a specified number of inputs, outputs, blocks, terminals per cell, and Rent exponent. However, the Rent exponent is treated as a target value that the program aims at and the synthetic benchmark circuits only follow Rent's rule on average, thus losing some of the controllability advantages. The `rmc` program also shows some other drawbacks, resulting from the hierarchical top-down approach that is followed (first, interconnections are laid out at the highest level; only at the end, connections are made between simple gates).

In [4], Stroobandt *et al.* presented a benchmark generation method, called `gnl`, that is also based on Rent's rule but that uses a bottom-up approach. This enhances the control on the various parameters. More importantly, and unlike Darnauer and Dai's method, the program also ensures a viable net degree distribution. The number of gates, the number of primary in- and outputs, and the Rent exponent can be predefined and the terminals-per-block distribution can be specified. A thorough theoretical deduction of several restrictions on the input parameters ensures that the program will eventually find a solution that obeys both Rent's rule and that results in the desired net degree distribution. Hence, a maximum amount of controllability is obtained. The basic method is reviewed in Section IV and is extended to allow inclusion of timing characteristics in Section V.

III. Characteristic Circuit Parameters

We want to model the partitioning properties of circuits, i.e., we want to have a notion of their interconnection complexity. This interconnection complexity is reflected in the Rent exponent and the net degree distribution. For a clear understanding, we start this section with an overview of the basic definitions used and we continue with a discussion on the Rent exponent and the net degree distribution.

A. Definitions

A circuit can be represented by a set of interconnected blocks as in Fig. 1(a) (the blocks can be the representation of transistors, gates, or even entire circuits). An interconnection between blocks is called a *net.* A net that is connected to more than two blocks is called a *multiterminal net.* We assume that a net cannot have more than one connection to the same block.[3] Some of the nets are also connected to the outside of the circuit. These nets are called *external nets* (as opposed to the *internal nets* which only connect blocks within the circuit). In order to model these external nets properly, we introduce a new kind of block which we call a *terminal* (of the circuit). The other blocks are called *logic blocks.* Every external net is connected to exactly one terminal. The *net degree* of a (multiterminal) net is defined as the number of blocks (logic blocks and circuit terminals) the net is connected to. A *net degree distribution* is a collection of values, indicating, for each net degree n, how many nets have a net degree equaling n.

Partitioning a circuit means dividing this circuit into disjoint subcircuits (called *modules*), each containing a subset of the blocks [Fig. 1(b)]. This partitioning is done using some kind of criterion. Generally, the criterion is to minimize the number of nets cut, i.e., the number of nets crossing the borders of modules in the partition. Recently, it has been observed that minimizing the number of terminals is a better criterion than minimizing the number of nets cut and that both criteria are not equal for multiterminal external nets [16]. Nets that are cut by module boundaries are shared between two or more modules and are said to be external to the modules. Therefore, the net is split into a number of subnets, one for each module that shares the net. A new terminal is assigned to each subnet (if the net was already external to the circuit then the terminal assigned to it can be reused for one of the subnets). Each module can then itself be seen as a circuit and can be partitioned further. A partitioning process where the modules themselves are recursively partitioned is called a *hierarchical partitioning method.*

[2] `rmc` is the acronym for "**r**andom **m**apped **c**ircuit"

[3] Otherwise, the two (or more) connected terminals can be grouped and treated as one terminal.

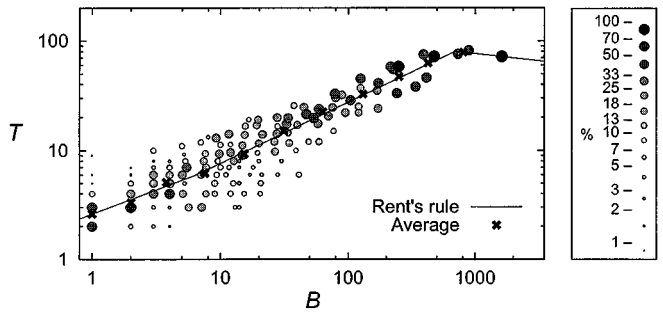

Fig. 2. Number of terminals versus number of blocks for every partition in the recursive partitioning (with `hMetis` [25]) of the ISCAS85 circuit "c3540nr," compared to Rent's rule. The size of the circles corresponds to the percentage of modules that has T terminals and B blocks in a pool of modules around an average number of blocks.

B. Rent's Rule: Interconnect Complexity Measure

Circuits can be classified on the basis of the notion that some circuits have a totally different structure of interconnections than others. These differences in *complexity* of the interconnect topology have been experimentally observed by Rent and his observations led to the well-known Rent's rule [14], a relationship between the average number of elementary blocks B in the modules of a partitioned circuit, and the average number of the module's external connections (terminals) T

$$T = tB^p \tag{1}$$

where t is the average number of terminals per logic block, and p is called the *Rent exponent.* This exponent is a measure of the interconnection complexity of the circuit. Its value is always smaller than one, with increasing values for increasing interconnection complexity. Generally, p ranges from 0.47 for regular circuits (such as Random Access Memories), up to 0.75 for complex circuits (such as fast full custom VLSI circuits) [17]. The validity of Rent's rule is a result of the fact that designers tend to build their circuits hierarchically, imposing the same complexity at each level of hierarchy. This leads to the observed "self-similarity" of circuits. Rent's rule can be observed in Fig. 2. The deviation of the data from the main power law of Rent's rule, observed for high values of T and B, is known as region II in Rent's rule [14], [18]. At the highest levels of the hierarchical partitioning method, the number of terminals (at the top level: pins) is lower than predicted by (1) due to the fact that designers have to deal with the pin limitation problem in today's chips. This effect is prevalent in most of today's designs and the pin limitation problem will probably not be solved anytime soon unless some very specific architecture changes become common practice, such as using area-I/O [19] or three-dimensional architectures [20], [21]. It has been observed that some circuits also show a deviation from Rent's rule for low values of T and B which can then be called region III [22]. As with region II, it is due to a boundary effect but it is of much less importance (in Fig. 2, region III is slightly visible as a crack in Rent's curve around $B = 8$).

C. The Net Degree Distribution

Another important parameter in characterizing circuits through their interconnection structure is the net degree of the nets. It has been observed that more than 75% of the nets in real circuits are two- and three-terminal nets [23]. A more elaborate study on multiterminal nets revealed that the distribution of net degrees generally follows a power law [24]. This power law distribution results from an accurate model of the behavior of multiterminal nets during the partitioning process. The model has been validated with benchmark data from the ISCAS[4] benchmark sets.

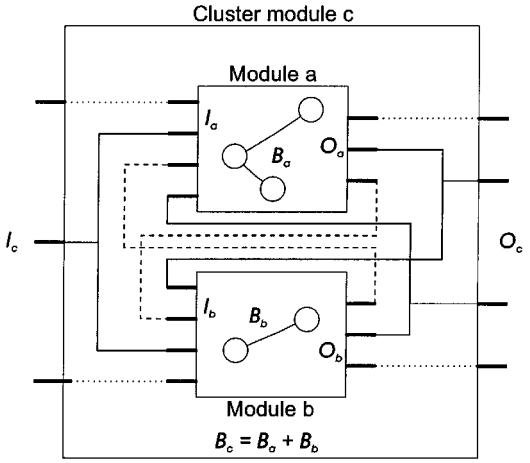

Fig. 3. Internal (dashed lines), external (solid lines), and pseudo-(dotted lines) connections combine modules a and b to cluster module c.

In addition to the requirement that benchmark circuits should obey Rent's rule, the power law net degree distribution should be found as well. Any synthetic benchmark circuit should at least have a net degree distribution that approximates the power law distribution in order to be a valid sample of real benchmark circuits.

IV. Benchmark Generation

For partitioning applications, the principal characteristic parameter to be taken into account is the interconnection structure. Therefore, the critical part of our benchmark generation method is the process of generating the netlist. We call this process the *net generation process.*

A. The Net Generation Process

The net generation process is explained using Fig. 3. Consider two modules a and b that are part of a certain partition of the benchmark circuit. We shall denote the number of logic blocks contained in those modules as B_a and B_b, respectively. The module that is formed by combining modules a and b, the *cluster module* c, then contains $B_c = B_a + B_b$ logic blocks. The numbers of inputs (I) and outputs (O) are denoted accordingly.

Basically, there are two types of connections possible between the modules a and b. The first type connects an output of one module with an input of the other module and does not leave the cluster module. We, therefore, call these *internal connections.* They are represented by a dashed line in Fig. 3. The other connections, the *external connections*, connect a terminal (input or output) of one module with an input of the other module

[4]Both the ISCAS85 and ISCAS89 benchmark sets were used.

and leave the cluster module through a terminal (solid lines). The terminals of the modules a and b that are not connected through an internal or external connection are routed directly to a terminal of the cluster module (dotted lines, these are not considered to be actual connections and are called "*pseudoconnections*"). We do not allow connections between two terminals of the same module since this type of connection can be made within the module itself (at a lower hierarchical level).

Our basic procedure for benchmark generation (where different net parts are connected) is the reverse of the partitioning process described in Section III (where nets are cut instead of combined). It is a bottom-up combination of logic blocks and can be described as follows.

1) All logic blocks in the circuit are generated and given the appropriate number of input and output terminals. The number of logic blocks and the number of inputs and outputs per logic block are specified by the user.
2) The logic blocks are paired and connections are made (randomly, but with certain restrictions) between their terminals. This results in a cluster of blocks with a number of input and output terminals.
3) The clusters themselves are recursively paired further with other clusters until all clusters are combined to one circuit.

Of course, the connections made in Step 2 of the generation process have to satisfy certain constraints in order to lead to a feasible benchmark circuit. First of all, the circuits must comply with the demand of a similar interconnection complexity as can be found in real circuits. Therefore, the number of terminals for the cluster module is defined by Rent's rule (1) $T_c = tB_c^p$. For circuits where the number of pins should be bounded, we can also introduce Rent's region II. Second, a power law net degree distribution should be obtained. For this, it suffices to aim at a constant ratio f of the number of internal connections to the total number of connections (pseudoconnections are not counted), at every level of the net generation method [4], [13]. Alternatively, one can compute the ratio of output terminals to the total number of terminals as a function of the module size $g(B)$ under the assumption of a constant f and try to obtain the right $g(B)$ [13].[5] This is the approach we use in this paper.

The constraints lead to restrictions on the choice of the number of connections from different types and can be reduced to necessary conditions on the circuit parameters (see [13]). If these conditions are met, a solution is guaranteed that obeys Rent's rule (on all hierarchical levels) and the desired power law net degree distribution. This guarantee can be given beforehand, preventing us from losing time trying to generate a circuit that is not feasible at all.

B. Rent's Region II

Although the number of gates within chips continues to increase and the pin limitation problem (and, hence, also region II) moves up to higher regions, the problem does not diminish, on the contrary. Realistic circuits will, therefore, by necessity, still have a strong region II. Being able to mimic this behavior in the synthetic circuits is then not only desirable, it is often necessary. For certain applications, e.g., for testing the interconnection structure of field programmable gate arrays, we wish to use circuits with a highly complex interconnection structure (high Rent exponent). The number of pins, on the other hand, should be limited to prevent large cell underutilization because of pin limitation problems (too high a number of pins limits the number of cells that can be effectively used for logic functions). The possibility of imposing a region II on the circuit is, therefore, an important feature.

Our bottom-up approach allows us to easily impose region II and, if desirable, region III on the synthetic circuit (in fact, the extension to an arbitrary number of regions is straightforward). It is sufficient to split the procedure in several parts. The first part remains the same as before. In a new region, the net generation process is to be seen as a new generation process starting from modules corresponding with the clusters found from the clustering in the previous region. But, this time, a new Rent exponent is used (resulting in a different slope for the T versus B curve).

V. Adding Functionality to gnl

In order to extend the scope of applicability for our benchmark circuits to timing-aware methods and tools, we have to include functionality to the circuit by allowing the use of a user-defined library of cells. The number of each type of library cells can be chosen and the choice of flip-flops or latches enables the generation of sequential circuits. In our previous work, synthetic benchmark circuits were only aimed at partitioning applications and, therefore, they were merely treated as (directed) graphs. However, using the benchmark circuits in logic optimizer tools (among others), we have to ensure that: 1) a net cannot be driven twice (this is already present in the method presented in Section IV; as a consequence, we do not allow bidirectional nets) and 2) no combinational loops are introduced during the net generation process. Therefore, a combinational loop prevention scheme is added that changes Step 2 of the net generation procedure as follows (Fig. 3):

- For each input terminal of the modules a and b, a list is compiled of all output terminals where the input value is observable through a combinational path. We call this list the *through-list.*
- Before a connection is made (e.g., between an output of module a and an input of module b), the through-list of the module b input is checked against the occurrence of the module a output.[6] If the output is not controllable by the input, the connection can be made, otherwise the connection is refused.
- If a connection is made, all through-lists are updated to the new situation.
- If all connections in the cluster module c are laid out, the through-lists are updated as the through-list for the complete module c.

[5]Since we are interested in obtaining the right ratio of output pins to total number of pins at the highest hierarchical level, this approach leads to better results if, mostly for discretization reasons or because a particular number of terminals deviates too much from the average, the goal cannot be met at some hierarchical levels.

[6]The through-list of the module b input could contain the module a output as a result of previously made connections.

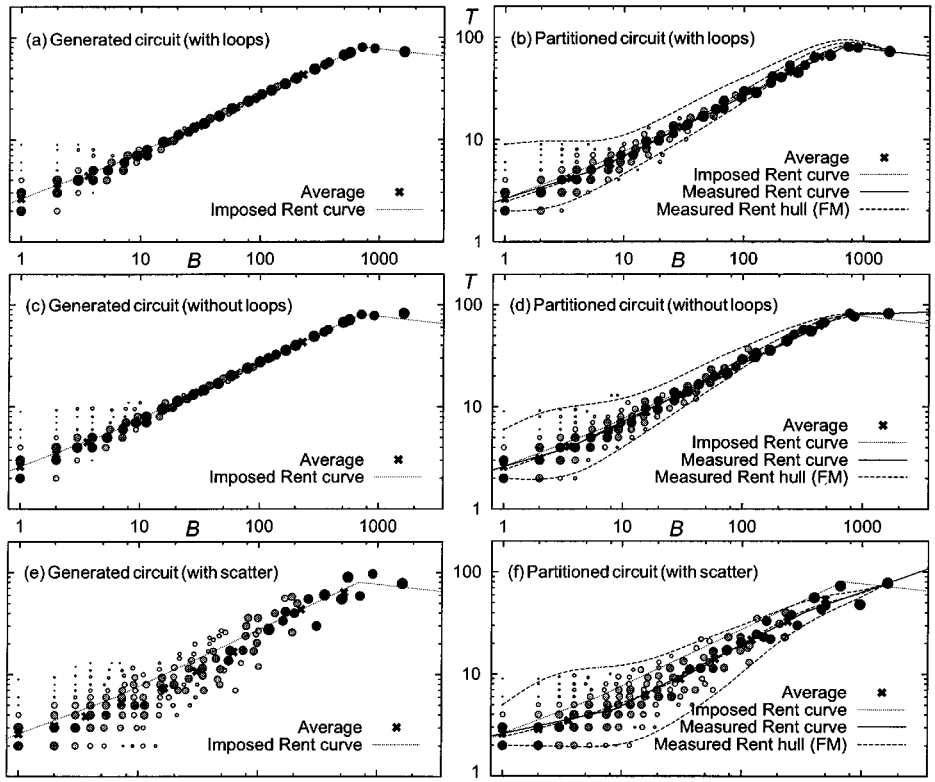

Fig. 4. Characteristic Rent curve for the synthetic circuits with (top) and without (middle) combinational loops and with imposed variation in the Rent curve (bottom). Imposed characteristic Rent curve (left) versus the one measured after recursive partitioning (right).

At the lowest level, the through-list of a logic gate is easily obtained. For the general cell types (AND, OR, NOT, NAND, NOR, XOR), all input through-lists contain all outputs. For flip-flops or latches, all through-lists are empty since there is no combinational path between input and output. Using the scheme above, the through-lists are known at every hierarchical level.

Note that the prevention of combinational loops is more than just a simple check. It significantly changes the way in which connections can be made. In particular, combinational loop prevention further restricts the connection possibilities and the restrictions depend on the choice of previous interconnections and the order in which they are chosen. In Section VI, we evaluate the influence of this addition and we show that the restrictions only marginally change the graph-based properties of the synthetic benchmark circuits, with the exception of the situation in which region II is prominent. The results show that synthetic benchmark generation for timing-aware tools becomes viable.

VI. Experimental Issues

In order to check that our (extended) benchmark generation method produces circuits with properties comparable to those of real circuits, we generate synthetic benchmark circuits based on the parameters of the ISCAS [1] and ISPD98 [2] benchmark circuits and compare the resulting circuits with the original ones.

A. Rent Curve

The net generation process induces a characteristic Rent curve in the synthetic benchmark circuit. This process corresponds to one single partitioning instance of the circuit, i.e., the reverse of the generation process. We have to check whether a similar characteristic Rent curve is observed for another partitioning instance (preferably the one leading to an optimal partitioning result). From Figs. 2 and 4, the characteristic Rent curve can be compared for the ISCAS85 benchmark circuit "c3540nr" and its synthetic counterparts. The results for the other benchmark circuits are similar. The distribution of terminals per logic block has been chosen exactly as in the original benchmark circuit. Fig. 2 shows the characteristic Rent curve of the original benchmark circuit after recursive bipartitioning with `hMetis` [25], Fig. 4(a) shows the characteristic Rent curve imposed by the generation program `gnl` (allowing combinational loops), and Fig. 4(b) shows the characteristic Rent curve of the synthetic benchmark circuit after recursive partitioning. Note that Fig. 4(a) follows the imposed Rent's rule almost perfectly (not taking the discretization effect into account).[7] A recursive partitioning of the synthetic circuit still gives a very acceptable characteristic Rent curve [Fig. 4(b)]. In any case, the characteristic Rent curve obtained through recursive partitioning is comparable for the original benchmark circuit (Fig. 2) and its synthetic counterpart [Fig. 4(b)].[8]

Fig. 4 also shows the Rent plot for the synthetic benchmark circuit without combinational loops (plots in the middle). The generation program `gnl` is still able to produce a benchmark

[7]Only at the lowest levels, there is a deviation from the desired characteristic Rent curve due to an excessive number of terminals for some logic blocks. More details can be found in [13].

[8]The scaling trend clearly remains visible in Fig. 4(b), a sign of the fact that the obedience to the Rent relation is visible in all parts of the synthetic circuit and is not a mere consequence of an imposed characteristic in some discrete points.

circuit with almost perfect characteristic Rent curve despite a severely limited choice for connections. The small deviation for region II is explained further. After recursive partitioning, the original Rent curve is still found. Using recursive partitioning schemes inferior to `hMetis`, we even find that the characteristic Rent curve matches the imposed Rent curve better when combinational loops are prevented than when they are allowed. This is to be expected since the prevention of combinational loops distributes the interconnection complexity more evenly over the hierarchical levels. Indeed, creating many loops within a hierarchical level increases the interconnection complexity difference between the "optimal" cut and an "inferior" one since a loop that is cut introduces two cuts instead of just one. Therefore, if no loops are present, the recursive partitioner has an easier job of finding a good cut, even if it did not find the absolutely best one. This effect is not observable with the `hMetis` partitioner in Fig. 4 because of its excellent quality. Therefore, we partitioned the circuits in this figure also with the Fiduccia–Mattheyses (FM) [26] partitioner and plotted the convex hull of the average maximal and minimal values, as well as the average value. Fig. 4(b) and (d) clearly shows that the FM partitioner results follow the imposed Rent's curve more closely in the case where no combinational loops are allowed.

Looking at Fig. 4, one also might remark on the homogeneity of the resulting benchmark circuits, especially with respect to obeying Rent's rule. We all know that real benchmark circuits never are that homogeneous as our synthetic circuits [27]. The circuits can be made more inhomogeneous by introducing a scatter around the value for the number of terminals predicted by Rent's rule. This is illustrated in Fig. 4(e) (generated circuit without combinational loops, but with scatter) and in Fig. 4(f) (same circuit after recursive partitioning).[9] Although Rent's curve for the synthetic benchmark circuits might look more realistic with the scatter, it is not clear whether we want to always use the possibility of scattering. One should weigh the gain in inhomogeneity against the loss of circuit parameter controllability, one of the main assets of synthetic benchmark circuits.

Comparing the three plots on the right side of Fig. 4, it seems that the middle one (without combinational loops) follows the imposed Rent's curve best. Two different phenomena seem to alter the Rent curve: 1) allowing combinational loops and 2) imposing a scatter on the imposed Rent behavior. However, the effect of allowing loops clearly does not lead to a more realistic solution. Indeed, Fig. 4(a) shows that the generated Rent curve is still very similar to the situation were combinational loops are prohibited [Fig. 4(c)] which indicates that the different Rent behavior after partitioning is only due to the fact that the interconnect complexity is imposed at some very specific points, with somewhat different complexity in intermediate ranges (due to the loops). This is not comparable to the situation in real circuits. The effect of scatter, on the other hand, really mimics the realistic behavior [Fig. 4(e)] and results in a Rent plot that is more spread out (as is the case in realistic circuits). This thus leads to the conclusion that the combinational loop prevention techniques presented in this paper do make the resulting circuits more realistic (compared to the situation with loops) but that introducing scatter can further improve them. We will return to this issue when we discuss the indirect validation.

[9]The scatter was introduced by adding multiplicative noise on the number of terminals during the generation process. Since the local Rent exponent cannot exceed one, the highest terminal counts could not be realized. As a result the average terminal count is lower than intended. This explains the deviation from the imposed Rent curve in Fig. 4(e).

B. Region II in Rent's Rule

If combinational loops are allowed, the imposed characteristic Rent curve with intentional region II can be easily reached by `gnl`. However, with combinational loop prevention enabled, a lower bound seems to exist[10] on the number of circuit pins that can be reached [compare Fig. 4(c) to Fig. 4(a)]. Experiments have shown that this lower bound depends mostly on the following three parameters: the Rent exponent for region II (or, alternatively, the intended number of primary pins), the number of flip-flops relative to the total number of circuit blocks, and the fraction of primary outputs to the total number of pins. Other parameters, such as the circuit size, only have an insignificant influence on the bound. Next, we try to evaluate the influence of the three parameters.

Fig. 5(a) shows the Rent plot for synthetic benchmark circuits with different instances of Rent's region II, in the case that loops are allowed. The intended characteristic Rent curve is easily reached. However, the restriction that no combinational loops are allowed influences this Rent curve. For a very prominent region II, a lower bound exists on the number of circuit pins, as can be observed from Fig. 5(b). The reason for this is to be found in the (negative) Rent exponent for region II being very low, which implies that a lot of pins have to be eliminated in the generation process by making connections. The number of new connections though is limited by the requirement of absence of combinational loops.

The value of the lower bound depends on how difficult it is to make connections without producing a combinational loop. To demonstrate this, we increase the relative number of flip-flops in Fig. 5(c) (for the total number of logic blocks remaining the same) for synthetic benchmark circuits with a very prominent region II. One can see that, when combinational loops are allowed or when the number of flip-flops is very high—thus, generating combinational loops is almost impossible—the desired number of pins can be reached. However, if no combinational loops are allowed and the number of flip-flops is low, the number of pins quickly reaches a lower bound.

The hardness of the lower pin bound also depends on the imposed ratio of output pins to the total number of pins (the *fraction g* of output terminals) [Fig. 5(d)]. A very high fraction of primary outputs causes more outputs to be present at the intermediate modules. As a result, more output pins have to be eliminated in the final stages of the net generation process. It is generally easy to eliminate inputs by merging them together (this will never introduce a loop). Outputs, on the other hand, can only be eliminated by connecting them to inputs, which is harder because of combinational loop prevention. When a relatively high number of output pins is desired, it is impossible to

[10]There is reason to believe that this bound is not a limitation induced by `gnl`, but that it is a fundamental restriction of homogeneous circuits.

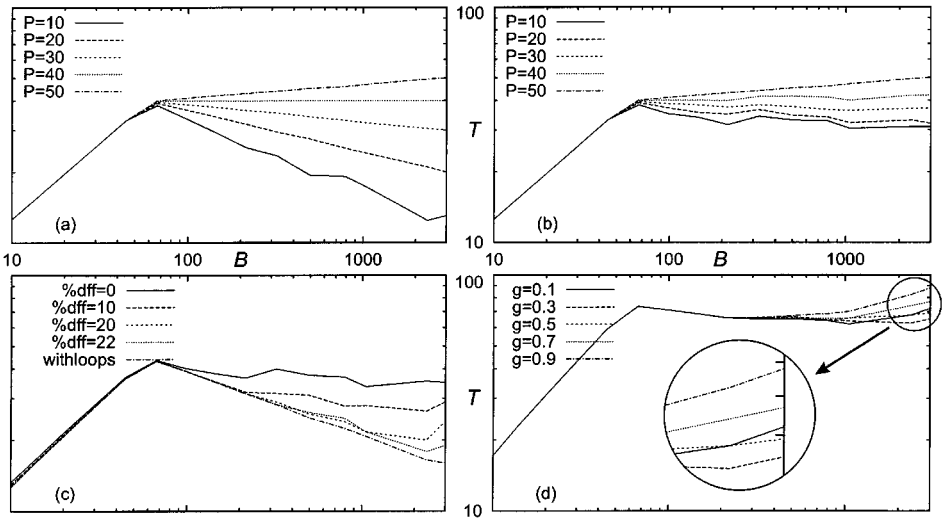

Fig. 5. Average characteristic Rent curve for synthetic benchmark circuits with different number of pins [both with (a) and without (b) loops], different number of flip-flops (c), and different number of outputs (d).

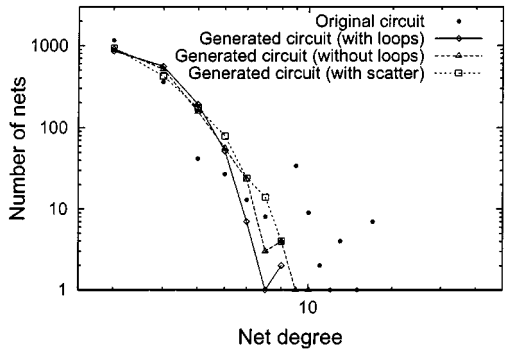

Fig. 6. Comparison between the net degree distribution of the benchmark circuit "c3540nr" and its synthetic counterparts with and without combinational loops and with scatter.

eliminate quite a few output pins, resulting in a higher number of circuit pins. In Fig. 5(d), we can observe that a very low fraction g also increases the total number of pins achievable. This is due to the fact that at the highest levels in the net generation process, when very few modules remain to be combined, the number of inputs and outputs can differ significantly because of the different module sizes. It is not uncommon that, once the output-to-input connections are made, the smallest module does not have enough inputs left to be combined with inputs of the larger block. At that point it becomes hard to eliminate inputs as well.

C. The Net Degree Distribution

Although the theoretical evaluation of the benchmark generation method [13] revealed that the net degree distribution for the synthetic benchmark circuits converges to a power law distribution, the question remains whether this property still holds for synthetic benchmarks that are free of combinational loops. Therefore, in Fig. 6, the net degree distribution of the original benchmark circuit "c3540nr" is compared to that of the synthetic benchmark circuits (with and without loops). All distributions more or less follow the same path around the expected power law distribution but the variations are larger in the real circuit, the reason being that real circuits are less homogeneous than our synthetic ones. Combinational loop prevention, as well as scatter in the Rent's curve, seem to have an insignificant influence on the net degree distribution.

D. Indirect Validation

Apart from the direct validation of the synthetic benchmark circuits by comparing their Rent curve and net degree distribution to those of real circuits, an indirect validation is equally important and should be performed as well. Since, at this moment, the synthetic benchmarks are intended for evaluating partitioning applications, the synthetic circuits are only viable alternatives for real circuits if these applications obtain comparable results for both circuits. In order to verify this, we cloned the ISPD98 benchmark suite [2] by generating synthetic counterparts with similar Rent characteristics and logic block degree distribution. For combinational loop prevention we assumed that 10% of the logic blocks are sequential. Then, we bipartitioned the circuits of the original and synthetic suites with several partitioning algorithms (`hMetis` [25], FM [26], iterative deletion (ID), ID postprocessed with FM (ID-FM) [28], and `ratiocut` [29]). Except for the latter the cost objective was "mincut" (minimal nets cut) with a minimal allowed imbalance. The `ratiocut` algorithm uses the "ratiocut" cost function which is the ratio of the minimum cut over the product of partition sizes. This cost function inherently favors equally sized partitions and hence does not require extra imbalance constraints. It does, however, allow quite imbalanced solutions, where balanced solutions are guaranteed for the other algorithms.

The results are shown in Fig. 7. In this figure, the benchmarks are ordered in the following way: for every partitioning algorithm, they are ordered in increasing cut size and then the best ordered ones over all algorithms are taken first. This provides smoother curves and an easier observation of the plots. Note that these figures are not intended to compare the different algorithms against each other. The important thing to observe here is that if one algorithm finds a better cut than another one for the

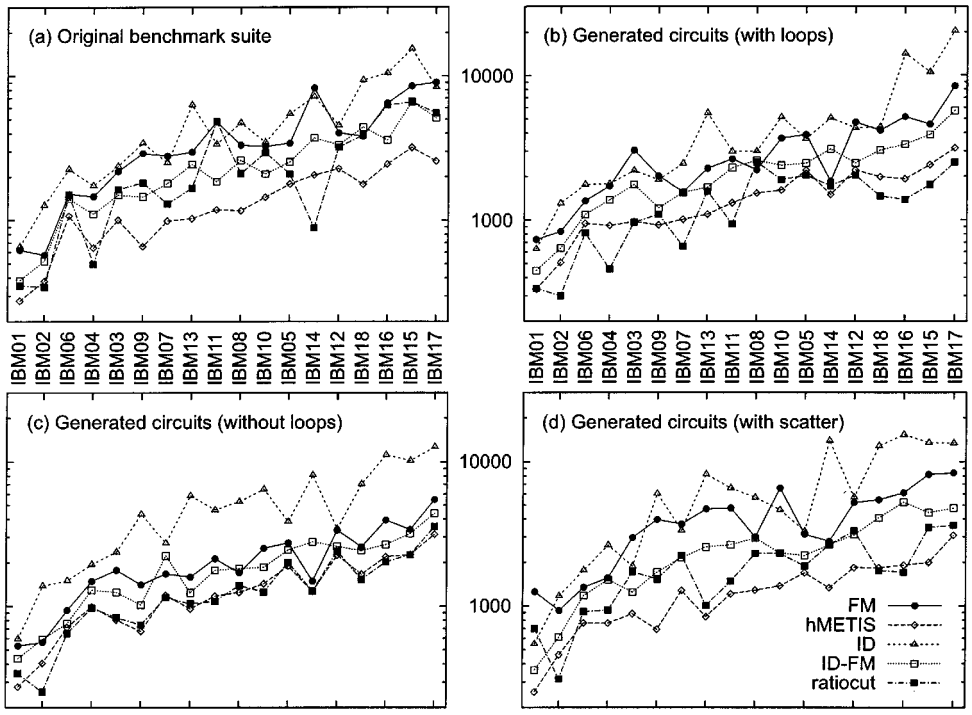

Fig. 7. Cut size using various bipartitioning algorithms, for: (a) the original ISPD98 benchmark circuits, and their synthetic counterparts, (b) with combinational loops allowed, (c) without combinational loops, and (d) with scatter in the imposed Rent curve.

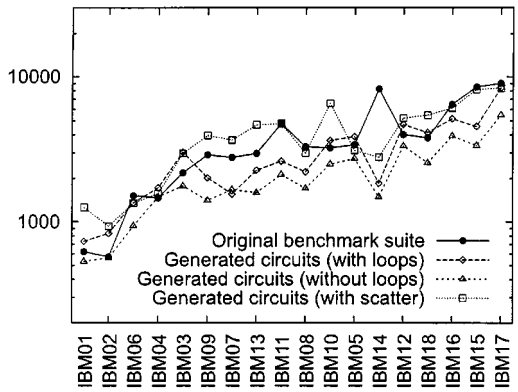

Fig. 8. Cut size using the FM algorithm, for the original ISPD98 benchmark circuits and their synthetic counterparts.

original circuit, it should also find a better cut for the synthetic counterpart.

The similarity we observe in the trends and the relative positions of the curves indicates that, at least qualitatively, the synthetic circuits do have a comparable interconnection structure as the original circuit. Of course, there are significant quantitative differences between the plots. Except for the inferior Iterative Deletion method, all partitioning tools seem to find very similar solutions for the synthetic benchmarks without combinational loops, whereas they did not for the case where loops are allowed and certainly not for the original benchmarks. The `hMetis` algorithm reports similar cut sizes for the original circuit and all synthetic counterparts, which indicates that the optimal cut size is similar in all cases. This is not the case for the other algorithms, as shown in Fig. 8 for the FM algorithm. Preventing combinational loops seems to have a positive impact on the optimality of the algorithm. The explanation for this phenomenon is to be found in the very nature of allowing combinational loops. As explained previously, the introduction of loops makes the difference between the "optimal" cut and the "inferior" cuts more prevalent. This increases the difference between local minima and the global minimal solution and hence makes the problem harder for the partitioning tools. Removing the possibility of having loops results in all tools finding cut sizes that are closer together, even though these might not be the optimal cut size at all.

The above explains the difference between the figures for the synthetic benchmark circuits with and without combinational loops. However, it does not explain the fact that all tools found very different results for the original benchmark since the original also does not contain combinational loops. This difference is, in our opinion, the result of the very high homogeneity in our synthetic circuits. Indeed, increasing the inhomogeneity in our generated circuits by adding a scatter to the imposed Rent behavior, tends to diversify between the algorithms [Fig. 7(d)]. It actually produces results that are now also quantitatively comparable to the original benchmark results (see also Fig. 8). Again, this leads to the conclusion that circuits where combinational loops are allowed are in fact less realistic than those without loops (although they falsely seem to be more realistic) and that introducing some inhomogeneities (in a controllable manner) makes the synthetic benchmarks more realistic.

Fig. 7 shows the same relative position of the curves for all plots, except for one: the curve for the `ratiocut` algorithm. While this algorithm performs mediocre on the original benchmarks, it is quite often the best for the synthetic benchmarks with loops, and one of the better for the synthetic benchmarks without loops. This is due to the fact that it allows imbalanced

solutions. For homogeneous circuits, the `ratiocut` algorithm performs quite well, and in some cases the resulting cut size is smaller than the result found by `hMetis`, which is known as the best algorithm. For the more realistic inhomogeneous circuits, `ratiocut` is clearly inferior to `hMetis`.

VII. Further Extensions for Timing-Aware Applications

The introduction of a functional identity to the logic blocks, together with the restriction that no combinational loops may be generated, not only makes our generation program applicable for evaluating a greater variety of CAD tools, it also results in more realistic circuits in general. However, some points need further attention: the possibly high degree of redundancy in our synthetic circuits and the introduction of more timing characteristics.

A. Redundancy in Synthetic Circuits

To evaluate the degree of redundancy in our synthetic benchmark circuits, we optimized all circuits with `SIS` [30], by invoking `script.rugged` and mapping the circuit with the minimum area criterion. We also optimized them using `FlowMap` [31].[11] We define the *redundancy factor* R as

$$R = 1 - G'/G \tag{2}$$

with G (G') the number of logic blocks or LUTs before (after) optimization. A comparison of the redundancy factor for the ISCAS89 circuits to the factor for the corresponding circuits generated by `gnl` (without combinational loops), is shown in Table I (R_o for the original circuits and R_s for their synthetic counterparts, both for optimizations using `SIS` and `FlowMap`). Since the assignment of the functionality to the gates in `gnl` is still done at random, we expect the generated circuits to be redundant [7]. The table shows, however, a much larger redundancy than we would imagine. The difference in redundancy between the original and synthetic circuits is expressed as the *relative redundancy factor* R_r defined as

$$R_r = 1 - G'_s/G'_o \tag{3}$$

where G'_s is the number of logic blocks or LUTs after optimization for the synthetic circuit and G'_o for the original one. Table I shows very high values for this relative redundancy factor R_r for both optimization tools.

It would be interesting to know whether or not (part of) the redundancy in synthetic benchmark circuits is due to circuit parameters. To investigate this, we examine the influence of the circuit parameters on the redundancy (we use `SIS` here). Only an insignificant correlation is found between the redundancy factor and the number of logic blocks in the circuit. However, we find a correlation between the redundancy factor and the Rent exponent [Fig. 9(a)].[12] The fact that the redundancy decreases with increasing interconnection complexity (increasing p) can be explained as follows: for circuits of low complexity (small p), the number of connections is relatively higher at the lowest partitioning levels, compared to complex circuits. Therefore, the chances of introducing reconvergence at low levels is higher for small values of p. Reconvergence is a prerequisite for redundancy, and since it is primarily the reconvergence at low levels (small sized modules) that has the most significant impact on redundancy (at higher levels the functionality of the modules is too complex), the circuits of low complexity have a higher chance to be redundant. This also explains why the Rent exponent in region II is not correlated to redundancy, which was also observed: region II only occurs at high partition levels. Although different Rent exponents in region II imply a different amount of reconvergence in the circuits, this has no significant influence on redundancy since the functionality of the modules is too complex at that level.

Another interesting observation is shown in Fig. 9(b), where we varied the relative number of flip-flop elements and observed that the redundancy factor decreases for an increasing number of flip-flops. This is a result of the fact that highly sequential circuits (more flip-flops) automatically reduce redundancy by breaking up (possibly reconvergent) combinational paths.

For lowering the redundancy, the adaptation of the circuit parameters (choosing a higher interconnection complexity or

TABLE I
Comparison of the Redundancy Factor R Between the Original ISCAS89 Benchmark Circuits (R_o) and Their Synthetic Counterparts (R_s) for Optimization with SIS and FlowMap. The Relative Redundancy Factor R_r Shows That the Synthetic Circuits Are a Lot More Redundant Than the Original Ones

Benchmark circuits		SIS			Flowmap		
Circuit	#cells	R_o (%)	R_s (%)	R_r (%)	R_o (%)	R_s (%)	R_r (%)
s27	13	0.00	92.31	92.31	38.46	46.15	12.50
s208.1	112	36.61	96.43	94.37	71.43	80.36	31.25
s344	175	27.43	89.71	85.83	59.43	83.43	59.15
s349	176	27.27	93.75	91.41	59.66	82.39	56.34
s382	179	22.91	98.88	98.55	55.87	84.92	65.82
s386	165	41.21	98.79	97.94	63.64	90.91	75.00
s400	186	21.51	92.47	90.41	56.45	91.94	81.48
s420.1	234	37.18	97.01	95.24	73.93	86.75	49.18
s444	202	29.70	99.01	98.59	59.90	91.58	79.01
s510	217	13.36	98.62	98.40	46.54	77.42	57.76
s526	214	21.96	98.60	98.20	66.36	97.20	91.67
s526n	215	18.60	96.74	96.00	63.72	97.21	92.31
s641	398	63.82	96.73	90.97	74.62	86.93	48.51
s713	412	65.05	97.33	92.36	75.49	91.99	67.33
s820	294	26.19	98.98	98.62	50.00	52.04	4.08
s832	292	22.95	98.97	98.67	48.63	61.99	26.00
s838.1	478	37.87	97.91	96.63	74.06	85.56	44.35
s953	424	22.88	89.15	85.93	46.93	42.69	-8.00
s1196	547	22.12	98.54	98.12	51.92	57.77	12.17
s1238	526	18.06	90.30	88.17	45.63	42.40	-5.94
s1423	731	15.60	99.59	99.51	61.56	97.26	92.88
s1488	659	30.35	99.70	99.56	49.77	96.66	93.35
s1494	653	29.25	99.69	99.57	48.70	96.94	94.03
s5378	2958	56.73	98.58	96.72	76.10	95.37	80.62
s9234.1	5808	82.71	99.85	99.10	89.96	99.17	91.77
s13207.1	8589	70.17	98.70	95.63	82.44	96.75	81.50

[11]Because this tool maps to field programmable gate array lookup tables (LUTs), we had to choose a different target library. As a result, the circuits seem more redundant.

[12]For the most complex circuits, `SIS` found an optimized circuit that contains more logic blocks than the original one, hence the negative redundancy factor. This phenomenon is due to the fact that `SIS` optimizes circuits to a generic library and maps this intermediate result afterwards to the specified library.

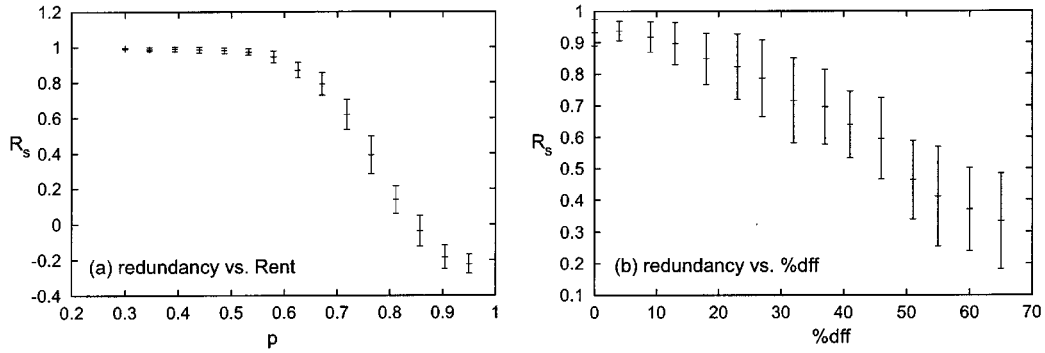

Fig. 9. Redundancy factor R_s for synthetic benchmarks as a function of the Rent exponent (a) and the number of flip-flops (b). Both the average value and the standard deviation for a set of generated benchmarks with the same parameters is shown.

a higher relative number of flip-flops) is a viable solution. Of course, we should aim at methods that inherently take redundancy into account. One approach could be to check if two (or more) gates share all of their input terminals. Such cases should be prohibited to prevent the obvious redundancy. A more fundamental way of preventing redundancy would be to check if reconvergent paths are introducing redundancy in a similar way as has been done for preventing combinational loops (by keeping through-lists that check the functionality of the modules). More research is needed in this respect.

B. Path Length Distribution

In order to obtain synthetic benchmark circuits that are really useful for timing-driven applications, more timing characteristics should be taken into account. One of the most important ones is to control the path length in the synthetic circuits. Although a solution to controlling path lengths is subject to further research, we believe a number of possibilities arise. We could try to break up long paths in a postprocessing step by adding flip-flops. However, this will change the structure of the circuit significantly and alleviates a lot of the problems with combinational loops and redundancy after they have to be dealt with. A better approach would aim at limiting the path length during the generation process. We believe that a similar approach as we use for the prevention of loops, by using through-lists, could also be used to keep track of path lengths. If such a list could, for instance, keep track of the longest path from the terminal to a flip-flop then we could prevent connections for which the combined paths exceed a threshold. Further research is planned to investigate these possibilities.

VIII. Conclusion

In order to broaden the scope of synthetic benchmark circuits to the evaluation of timing-driven or logic optimizer applications, functionality has to be included. We extended our existing graph-based benchmark generation method to allow a user-defined library cell selection, together with a method for preventing combinational loops. Although the number of possible connections is restricted by prohibiting combinational loops, experiments show that this has an insignificant or even positive influence on both the resulting characteristic Rent curve and the net degree distribution, which are the principal interconnection parameters for our graph-based generation method. Both a direct validation of these principal parameters, as well as an indirect validation through the partitioning of both the original circuit and its synthetic counterpart showed that our generated benchmarks are viable alternatives for real circuits. The specifics of timing-driven CAD tools require further extensions to the method but we have shown that such extensions become feasible and that our method has the potential to be useful for timing-aware tools.

Acknowledgment

The authors wish to thank P. H. Madden, G. Karypis, and V. Kumar, and S. Dutt, W. Deng, and C. Alpert for making their partitioning programs available.

References

[1] Computer-Aided Design Benchmarking Laboratory [Online] Available: http://www.cbl.ncsu.edu/benchmarks/

[2] C. J. Alpert, "The ISPD circuit benchmark suite," in *Proc. ACM/SIGDA Int. Symp. Physical Design*, Apr. 1998, pp. 85–90.

[3] P. Verplaetse, J. Van Campenhout, and D. Stroobandt, "On synthetic benchmark generation methods," in *Proc. IEEE Int. Symp. Circuits and Systems*, May 2000, pp. IV-213–IV-216.

[4] D. Stroobandt, J. Depreitere, and J. Van Campenhout, "Generating new benchmark designs using a multiterminal net model," *Integration, the VLSI J.*, vol. 27, no. 2, pp. 113–129, July 1999.

[5] C. J. Alpert and A. B. Kahng, "Recent directions in netlist partitioning: a survey," *Integration, the VLSI J.*, vol. 19, no. 1–2, pp. 1–81, 1995.

[6] K. Iwama and K. Hino, "Random generation of test instances for logic optimizers," in *Proc. 31st ACM/IEEE Design Automation Conf.*, June 1994, pp. 430–434.

[7] K. Iwama, K. Hino, H. Kurokawa, and S. Sawada, "Random benchmark circuits with controlled attributes," in *Proc. Eur. Design Test Conf.*, 1997, Available: CD-ROM.

[8] D. Ghosh, N. Kapur, J. Harlow III, and F. Brglez, "Synthesis of wiring signature-invariant equivalence class circuit mutants and applications to benchmarking," in *Proc. Design, Automation, and Test in Eur. Conf.*, Feb. 1998, pp. 656–663.

[9] J. E. Harlow III and F. Brglez, "Synthesis of ESI equivalence class combinational circuit mutants," CBL, CS Dept., NCSU, Tech. Rep. 1997-TR@CBL-07-Harlow. Available: http://www.cbl.ncsu.edu/publications, Oct. 1997.

[10] M. D. Hutton, J. Rose, J. P. Grossman, and D. Corneil, "Characterization and parameterized generation of synthetic combinational circuits," *IEEE Trans. Computer-Aided Design*, vol. 17, pp. 985–996, Oct. 1998.

[11] M. Hutton, J. Rose, and D. Corneil, "Generation of synthetic sequential benchmark circuits," in *Proc. ACM/SIGDA 5th Int. Symp. Field Programmable Gate Arrays*, Feb. 1997, pp. 149–155.

[12] J. Pistorius, E. Legai, and M. Minoux, "Generation of very large circuits to benchmark the partitioning of FPGAs," in *Proc. ACM/SIGDA Int. Symp. Physical Design*, Apr. 1999, pp. 67–73.

[13] D. Stroobandt, "Analytical Methods for a priori Wire Length Estimates in Computer systems," Ph.D. dissertation, Ghent Univ., Faculty of Appl. Sci., Ghent, Nov. 1998.
[14] B. S. Landman and R. L. Russo, "On a pin versus block relationship for partitions of logic graphs," *IEEE Trans. Comput.*, vol. C-20, pp. 1469–1479, Dec. 1971.
[15] J. Darnauer and W. W. Dai, "A method for generating random circuits and its application to routability measurement," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, Feb. 1996, pp. 66–72.
[16] D. Stroobandt, "Pin count prediction in ratio cut partitioning for VLSI and ULSI," in *Proc. IEEE Int. Symp. Circuits and Systems*, May 1999, pp. VI-262–VI-265.
[17] R. L. Russo, "On the tradeoff between logic performance and circuit-to-pin ratio for LSI," *IEEE Trans. Comput.*, vol. C-21, pp. 147–153, Feb. 1972.
[18] H. Van Marck, D. Stroobandt, and J. Van Campenhout, "Toward an extension of Rent's rule for describing local variations in interconnection complexity," in *Proc. 4th Int. Conf. Young Computer Scientists*, July 1995, pp. 136–141.
[19] J. Depreitere, H. Van Marck, and J. Van Campenhout, "A quantitative analysis of the benefits of the use of area-I/O pads in FPGAs," *Microprocessors and Microsystems*, vol. 21, no. 2, pp. 89–97, Oct. 1997.
[20] D. Stroobandt and J. Van Campenhout, "Estimating interconnection lengths in three-dimensional computer systems," *IEICE Trans. Inform. Syst.*, vol. E80-D, no. 10, pp. 1024–1031, Oct. 1997.
[21] J. Van Campenhout, H. Van Marck, J. Depreitere, and J. Dambre, "Optoelectronic FPGAs," *IEEE J. Select. Topics Quantum Electron.*, vol. 5, no. 2, pp. 306–315, Mar./Apr. 1999.
[22] D. Stroobandt, "On an efficient method for estimating the interconnection complexity of designs and on the existence of region III in Rent's rule," in *Proc. 9th Great Lakes Symp. VLSI*, Mar. 1999, pp. 330–331.
[23] E. S. Kuh and T. Ohtsuki, "Recent advances in VLSI layout," *Proc. IEEE*, vol. 78, pp. 237–263, Feb. 1990.
[24] D. Stroobandt and F. J. Kurdahi, "On the characterization of multipoint nets in electronic designs," in *Proc. 8th Great Lakes Symp. VLSI*, Feb. 1998, pp. 344–350.
[25] G. Karypis and V. Kumar. (1998, Nov.) hMetis: A Hypergraph Partitioning Package. [Online]. Available: http://www-users.cs.umn.edu/~karypis/metis/hmetis/main.shtml
[26] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," in *Proc. 19th IEEE Design Automation Conf.*, Apr. 1982, pp. 175–181.
[27] P. Verplaetse, D. Stroobandt, and J. Van Campenhout, "A stochastic model for interconnection complexity based on Rent's rule," in *Proc. IEEE Int. Workshop Logic Synthesis*, June 2000, pp. 319–325.
[28] P. H. Madden, "Partitioning by iterative deletion," in *Proc. ACM/SIGDA Int. Symp. Physical Design*, Apr. 1999, pp. 83–89.
[29] Y.-C. Wei and C.-K. Cheng, "Ratio cut partitioning for hierarchical designs," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 911–921, July 1991.
[30] E. M. Sentovich *et al.*, "SIS: A System for Sequential Circuit Analysis," Univ. California, Berkeley, Tech. Rep. UCB/ERL M92/41, 1992.
[31] J. Cong and Y. Ding, "Flowmap: An optimal technology mapping algorithm for delay optimization in lookup-table-based FPGA designs," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 1–12, Jan. 1994.

Dirk Stroobandt (S'92–M'98) was born in Ghent, Belgium, in 1972. He graduated in 1994 as electrotechnical engineer at Ghent University and received the Ph.D. degree from the same university in 1998.

From 1994 to 1998, he was Research Assistant and since 1998, he is Post-doctoral Fellow with the Fund for Scientific Research—Flanders. From July 1999 to June 2000, he visited the University of California at Los Angeles as a Post-Doctoral Researcher, where he was affiliated to the group of A. B. Kahng. His research interests include the modeling of three-dimensional optoelectronic systems, estimating interconnection lengths in computer systems, and characterization of design parameters.

Dr. Stroobandt is the inaugural winner of the ACM/SIGDA Outstanding Doctoral Thesis Award (presented at DAC'99). In 1999, he initiated the Workshop on System-Level Interconnect Prediction (SLIP). He was the General Chair of SLIP 2000. He is a Member of AIG, KVIV, and ACM.

Peter Verplaetse (S'96) was born in Ghent, Belgium, in 1973. In 1996, he graduated as electrotechnical engineer at Ghent University. In 1997, he received the M.S. degree in electrical engineering from Stanford University, Stanford, CA. He is working toward the Ph.D. degree at the samne university.

Currently, he is a Research Assistant of the Fund for Scientific Research—Flanders. He is affiliated with the Department of Electronics and Information Systems at Ghent University. His research mainly focuses on interconnection complexity measures for VLSI circuits, and their relationship with functional complexity. His other research interests include CAD algorithms for VLSI circuit and asynchronous circuits.

Jan Van Campenhout (M'94) was born in Vilvoorde, Belgium, on August 9, 1949. He received a degree in electromechanical engineering from Ghent University, Ghent,Belgium, in 1972; and the M.S.E.E. and Ph.D. degrees from Stanford University, Stanford, CA, in 1975 and 1978, respectively.

He teaches courses in computer architecture, electronics, and digital design at the Faculty of Applied Sciences of Ghent University. His current research interests include the study and implementation of various forms of parallelism in computer systems, and their application in programming language support, computer graphics, and robotics.

Prof. Van Campenhout is currently the head of the ELIS Department in the Faculty of Engineering. He is a Member of Sigma Xi, KVIV, and ACM.