TOWARDS SYNTHETIC BENCHMARK CIRCUITS FOR EVALUATING TIMING-DRIVEN CAD TOOLS

Dirk Stroobandt[*] *Peter Verplaetse*[†] *Jan Van Campenhout*

University of Ghent, Department of Electronics and Information Systems, Belgium

{dstr, pvrplaet, jvc}@elis.rug.ac.be

ABSTRACT

For the development and evaluation of CAD-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 for use in evaluation tools, one could consider to actually generate such 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. However, the synthetic benchmark circuits are still very redundant, compared to existing sets of real benchmarks. It is shown that a correlation exists between the degree of redundancy and key circuit parameters.

1. INTRODUCTION

The production of 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 for which the CAD tool is aimed.

Initiatives for distributing benchmark circuits have been taken (ISCAS benchmarks, [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 ATPG (Automatic Test Pattern Generation) tools and, hence, contain special structures that might deviate from what can be expected in a general circuit.

New sets of benchmark circuits are definitely needed for evaluating new tools that are developed in several research groups. Only recently, synthetic benchmark generation is becoming to be recognized as a viable alternative. The major problem of synthetic benchmark generation is the requirement that, for the application the benchmark circuits are intended for, they are good representatives of real circuits, i.e., circuits that could be the result of a real design process.

The research community has tried to come up with different ways of generating random circuits (some of them are presented in [3], 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 based on applying a sequence of random transformations on an initial (existing) circuit [4, 5]. Hutton et al. [6] addressed the problem of random generation of circuits "from scratch". They defined properties such as size, delay, physical shape, edge-length distribution, and fanout distribution to describe the physical characteristics and generated circuits with an exact parameterization ("clones" of existing 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. 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

[*] Post-doctoral Fellow of the Fund for Scientific Research – Flanders (Belgium) (F.W.O.).

[†] Research Assistant of the Fund for Scientific Research – Flanders (Belgium) (F.W.O.).

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.

For the evaluation of algorithms that are related to partitioning, the interconnection complexity is the main characterization parameter. It is reflected through Rent's rule and the so called Rent exponent of the circuit [7] (see section 2). Darnauer and Dai [8] were the first to attempt to generate random benchmark circuits, based on Rent's rule. Their program, called `rmc`[1], 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 for 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 [9], Stroobandt 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 program `gnl` lets the user predefine the number of gates, the number of primary in- and outputs, and the Rent exponent, as well as the terminals-per-block distribution. 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 a power law net degree distribution. Hence, for the first time, a maximum amount of controllability is obtained while the graph-based properties of real circuits are maintained. The basic method will be reviewed in section 3.

In this paper, we seek to include timing properties in the evaluation procedure, since timing-driven partitioning and placement is a major concern these days. Whereas previous efforts for benchmark generation mainly focused on 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, placers need information on the block area, and routers need information on the terminal positions. It is therefore desired 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 behaviour at the logic level, hence the generated circuits should be free from combinational loops.

To obtain this goal, we extend our previous benchmark generation program `gnl`[2] [9] by enabling the choice of gates from a user-defined library. The inclusion of flip-flops also enables us to generate sequential circuits. Special care is taken to exclude connections that introduce combinational loops. The experimental results show that this does not fundamentally change the characteristic Rent curve of the generated circuit, nor the net degree distribution, two characteristic parameters of the graph-based model. 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. Although the results show a correlation of the degree of redundancy in synthetic benchmark circuits to the degree of redundancy in real benchmark circuits, the synthetic benchmark circuits are still more redundant than their original counterparts. Further extensions to the generation procedure are discussed to mitigate this problem.

[1] `rmc` is the acronym for "Random Mapped Circuit"

[2] `gnl` is the acronym for "Generate NetList"

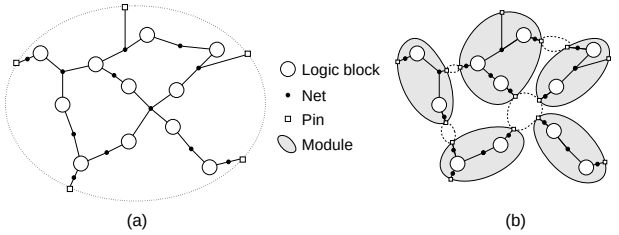

Figure 1. Model of a circuit (a) and the partitioning of the circuit into modules (b).

Section 2 gives an overview of the parameters that characterize real circuits and section 3 describes the basic procedure of generating benchmark circuits. The features of our new benchmark generation method will be the subject of section 4 whereas section 5 is devoted to some interesting experimental results. In section 6, some of the weaknesses of the proposed method are addressed and an indication is given on how solutions can be found in future work on this topic.

2. 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 will start this section with an overview of the basic definitions used and we will continue with a discussion on the Rent exponent and the net degree distribution.

2.1. Definitions

A circuit can be represented by a set of interconnected blocks as in figure 1(a) (the blocks can be the representation of transistors, gates, or even whole circuits). An interconnection between blocks is called a *net*. A net that is connected to more than two blocks is called a *multi-terminal net*. 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 will call a *pin*. The other blocks will be called *logic blocks*. Every external net will be connected to exactly one pin. The *net degree* of a (multi-terminal) net will be defined as the number of blocks (logic blocks and pins) 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 equalling n.

Partitioning a circuit means dividing this circuit into disjoint sub-circuits (called *modules*), each containing a subset of the blocks (figure 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. 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 will be split into a number of sub-nets,

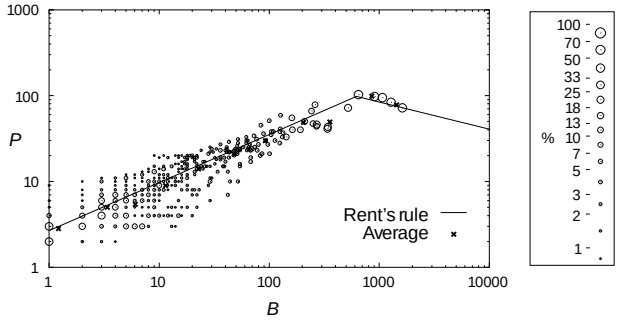

Figure 2. Number of pins versus number of blocks for every partition in the 'ratiocut' partitioning of the 'c3540nr' benchmark circuit, compared to Rent's rule. The size of the circles corresponds to the percentage of modules that has P pins and B blocks in a pool of modules around an average number of blocks.

one for each module that shares the net. A new pin will be assigned to each sub-net (if the net was already external to the circuit then the pin assigned to it can be reused for one of the sub-nets). 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 will be called a *hierarchical partitioning method*. The `ratiocut` partitioning method [10] is known to be one of the better hierarchical partitioning methods [11] and will be used throughout this paper.

2.2. 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 *interconnection complexity* have been experimentally observed by Rent and his observations led to the well-known Rent's rule [7], 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 (pins) P:

$$P = T_b\, B^r, \qquad (1)$$

where T_b is the average number of terminals per logic block, and r is called the *Rent exponent*. This exponent is a measure of the interconnection complexity of the circuit. Its value is always smaller than 1, with increasing values for increasing interconnection complexity. Generally, r 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) [12]. 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 figure 2. The deviation of Rent's rule from the data, observed for high values of P and B, is known as region II in Rent's rule [7, 13]. At the highest levels of the hierarchical partitioning method, the number of pins is lower than predicted by equation 1 due to the fact that designers have to deal with the pin limitation problem in today's chips.

2.3. 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 2- and 3-terminal nets [14]. A more elaborate study on multi-terminal nets revealed that the distribution of net degrees generally follows a power law [15]. This power law distribution results from an accurate model of the behaviour of multi-terminal nets during the partitioning process. The model has been validated with benchmark data from the ISCAS benchmark set.

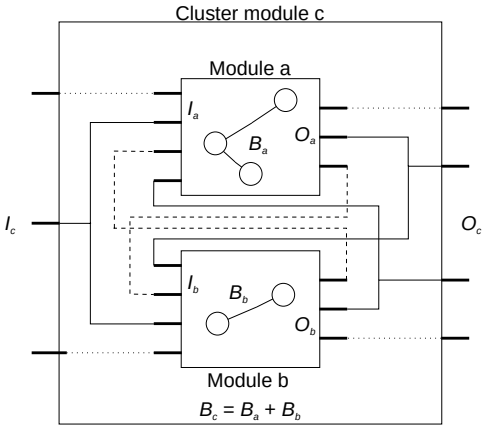

Figure 3. The net generation process.

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.

3. 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 will call this process the *net generation process*.

The net generation process will be explained using figure 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 will therefore call these *internal connections*. They are represented by a dashed line in figure 3. The other connections, the *external connections*, connect a pin (input or output) of one module with an input of the other module and leave the cluster module through a pin (full lines). The pins of the modules `a` and `b` that are not connected through an internal or external connection are routed directly to a pin of the cluster module (point lines, these are not considered to be actual connections and will be called "*pseudo-connections*"). We do not allow connections between two pins 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 2 (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 pins for the cluster module is defined by Rent's rule (equation 1) $P_c = T_b\, B_c^r$. For circuits where the number of pins should be bounded, we can also introduce Rent's region II. Secondly, a power law net degree distribution should be obtained. For this, it suffices to aim at a constant ratio of the number of internal connections to the total number of connections (pseudo-connections are not counted), at every level of the hierarchical partitioning method [9, 16].

The constraints lead to restrictions on the choice of the number of connections from different types (see [16]) and can be reduced to necessary conditions on the circuit parameters (see [16]). 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.

Rent's region II can be easily imposed. It is sufficient to split the procedure in two parts. The first part remains the same as before. In Rent's region II, the net generation process is to be seen as a new generation process starting from modules corresponding with the clusters found from the previous clustering in the normal region. But, this time, a new Rent exponent is used (resulting in a different slope for the P versus B curve in region II).

4. ADDING FUNCTIONALITY TO GNL

In order to extend the scope of applicability for our benchmark circuits, we include functionality 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 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 no combinational loops are introduced during the net generation process. Therefore, an optional parameter 'noloops' is added that changes step 2 of the net generation procedure as follows (figure 3):

- For each input terminal of the modules a and b, a list is compiled of all output terminals were 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.[3] 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.

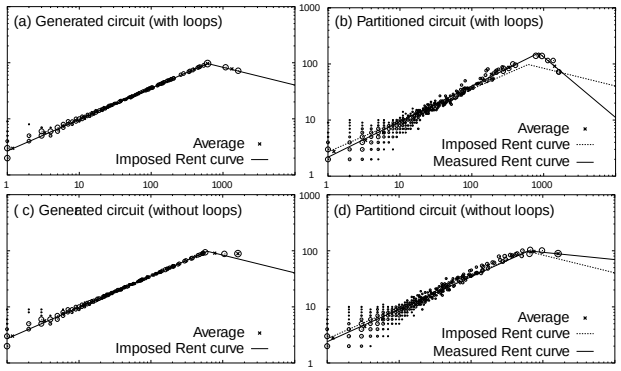

Figure 4. Characteristic Rent curve for the synthetic circuits with (top) and without (bottom) combinational loops. Imposed characteristic Rent curve (left) versus the one measured after ratiocut partitioning (right).

At the lowest level, the through-list of the logic gates is easily obtained. For the general cell types (AND, OR, NOT, NAND, NOR, XOR), all input through-lists contain all outputs. For flip-flops, all through-lists are empty since there is no combinational path between input and output.

Combinational loop prevention thus further restricts the connection possibilities and the restrictions depend on the choice of interconnections and the order in which they are chosen. In the next section, we will show that these restrictions do not fundamentally change the properties of the synthetic benchmark circuit, with the exception of the situation in which region II is prominent.

5. EXPERIMENTAL ISSUES

In order to check that our 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 benchmark circuits and compare the resulting circuits with the original ones.

5.1. 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 still have to check whether a similar characteristic Rent curve is observed for another partitioning instance (preferably the one leading to an optimal partitioning result). In figures 2 and 4, the characteristic Rent curve is 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

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

as in the original benchmark circuit. Figure 2 shows the characteristic Rent curve of the original benchmark circuit after partitioning with `ratiocut`, figure 4(a) shows the characteristic Rent curve imposed by the generation program `gnl`, and figure 4(b) shows the characteristic Rent curve of the synthetic benchmark circuit after partitioning with `ratiocut`. Note that figure 4(a) follows Rent's rule perfectly (not taking the discretization effect into account).[4] A partitioning of the synthetic circuit by `ratiocut` still gives an acceptable characteristic Rent curve (figure 4(b)) but the partitioning program did not find the optimal solution which results in a seemingly higher complexity (note that this is probably also the case in real circuits). In any case, the characteristic Rent curve obtained through `ratiocut` partitioning is comparable for the original benchmark circuit (figure 2) and its synthetic counterpart (figure 4(b)).[5]

Figure 4 also shows the Rent plot for the synthetic benchmark circuit with the 'noloops' option set (plots at the bottom). The generation program `gnl` is still able to produce a benchmark circuit with perfect characteristic Rent curve despite a severely limited choice for connections. After partitioning with `ratiocut`, the characteristic Rent curve matches the imposed Rent curve even better than when the 'noloops' option is not set (compare figure 4(d) to figure 4(b)). This could be expected since the 'noloops' option prevents combinational loops. Hence, the interconnection complexity is more evenly distributed over the hierarchical levels. Indeed, creating many loops within a hierarchical level increases the interconnection complexity difference between the "good" cut and a "bad" one since a loop that is cut introduces two cuts instead of just one. Therefore, if no loops are present, `ratiocut` has an easier job of finding a good cut, even if it did not find the absolutely best one. This observation seems to indicate that the introduction of the 'noloops' option in `gnl` automatically results in circuits with a more realistic characteristic Rent curve.

5.2. Region II in Rent's rule

If combinational loops are allowed, the intended characteristic Rent curve in region II is easily reached by `gnl`. However, with the 'noloops' option set, a lower bound seems to exist[6] on the number of circuit pins that can be reached (compare figure 4(c) to figure 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 valuate the influence of the three parameters.

Figure 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

[4] Only at the lowest levels, there is a small deviation from the desired characteristic Rent curve due to an excessive number of terminals for some logic blocks. More details can be found in [16].

[5] The scaling trend clearly remains visible in figure 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.

[6] 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.

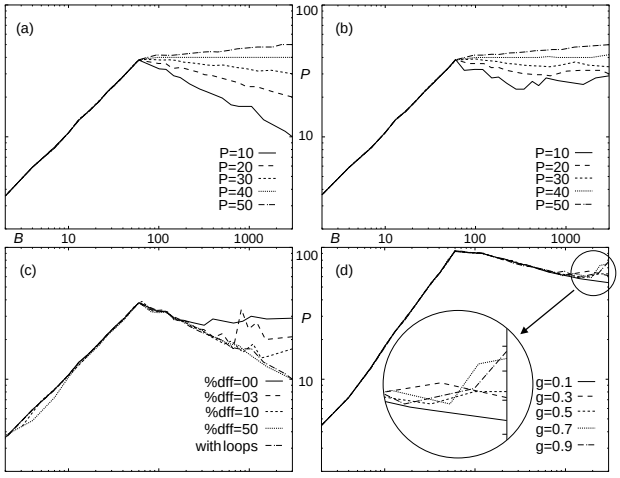

Figure 5. Average characteristic Rent curve for synthetic benchmark circuits with different number of pins (both with (a) and without (b) loops), different number of flip-flops (c), and different number of outputs (d).

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 figure 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 this 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 figure 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 loops is almost impossible— the desired number of pins can easily 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) (figure 5(d)). The reason is quite simply that the number of output terminals can only be reduced by making an internal connection (see figure 3). External connections do not change the number of output terminals. However, every connection reduces at least one input terminal. So, from the moment the imposed number of output terminals is reached, no connections can be made further without augmenting the ratio of outputs to terminals. It is clear that the problem is worse if a relatively high number of output pins is desired.

5.3. The net degree distribution

Although the theoretical evaluation of the benchmark generation method [9] 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 benchmarks generated with the 'noloops' option. Therefore, in figure 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 follow the same path but the vari-

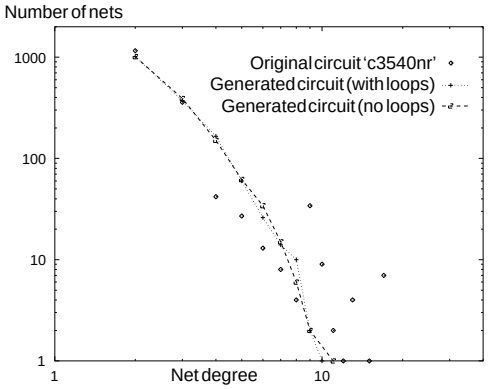

Figure 6. Comparison between the net degree distribution of the benchmark circuit 'c3540nr' and its synthetic counterparts with and without combinational loops.

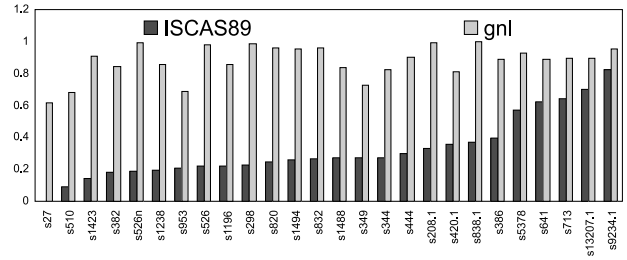

Figure 7. Comparison of the redundancy factor R_f between the original ISCAS89 benchmark circuits and their synthetic counterparts.

ations are larger in the real circuit, the reason being that real circuits are less homogeneous than our synthetic ones. The 'noloops' option seems to have an insignificant influence on the net degree distribution.

5.4. Redundancy in synthetic circuits

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. Only one point needs further attention: the possibly high degree of redundancy in synthetic circuits.

To evaluate the degree of redundancy in our synthetic benchmark circuits, we optimized all circuits with SIS, by invoking `script.rugged` and mapping the circuit with the minimum area criterion. We define the *redundancy factor* R_f as $R_f = 1 - G'/G$, with G (G') the number of logic blocks before (after) optimization. A comparison of the redundancy factor for the ISCAS89 circuits to the factor for the corresponding circuits generated by `gnl`, is shown in figure 7. Since the assignment of the functionality to the gates is still done at random, we expect the generated circuits to be redundant [4]. Indeed, it is quite simple to generate a completely redundant block. Consider, for instance, figure 8. The output `a` of a large module is doubled and reconverges into an NAND-gate again. One of the reconvergent connections is sent through an inverter gate first. The result is always 1, making the whole module redundant.

It would be interesting to know whether or not (part of) the redundancy in synthetic benchmark circuits is due to circuit pa-

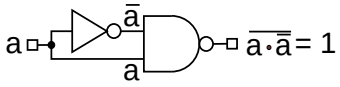

Figure 8. An example of a simple redundant circuit.

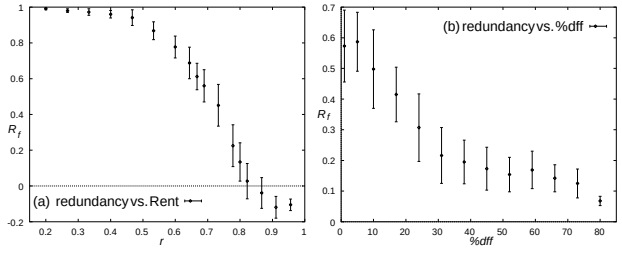

Figure 9. Redundancy factor R_f as a function of the Rent exponent (a) and the number of flip-flops (b). Both the average value and the standard deviation for a set of generated benchmarks with the same parameters is shown.

rameters. To investigate this, we examined the influence of the circuit parameters on the redundancy. No correlation was found between the redundancy factor and the number of logic blocks in the circuit. However, we did find a correlation between the redundancy factor and the Rent exponent (figure 9(a)).[7] The fact that the redundancy decreases with increasing interconnection complexity (increasing r) can be explained as follows: for circuits of low complexity (small r), 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 r. 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 since the functionality of the modules is too complex.

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

6. FUTURE RESEARCH

It is obvious from the results shown in the previous section that `gnl` still has a few shortcomings: the number of pins can deviate from the intended number if a prominent region II exists and no combinational loops are allowed; the high redundancy in the synthetic circuits; and the homogeneity of the synthetic circuits.

The generation procedure is aimed at following the characteris-

[7] 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.

tic Rent curve in the first place. To obtain the exact number of input or output pins as desired, many solutions could be considered. One possibility is to alleviate the problem by setting the number of output terminals at the boundary between the normal Rent region and region II low (ultimately, as low as the final number of output pins). The excess of terminals will then consist only of input terminals. Those can be reduced more easily which results in a lower bound on the minimal number of pins at the end. Another solution consists of keeping some logic blocks at hand, not including them in the generation process. In region II, these logic blocks can be used for reconverging two outputs into one. A third solution is to backtrace to smaller blocks and to gradually change the connections or the number of pins of lower modules to enable the parameter choices at the higher levels.

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

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. In fact, making the circuits more inhomogeneous is not a problem at all. This can simply be done by introducing a scatter around the value for the number of pins predicted by Rent's rule. Although Rent's curve for the synthetic benchmark circuits then might look more realistic, it is questionable if they really would be. One should weigh the gain in inhomogeneity against the loss of circuit parameter controllability.

7. 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 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 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.

One problem that remains to be solved is the redundancy within the new synthetic circuits. We showed that the redundancy factor decreases with increasing Rent exponent and with an increasing number of flip-flops. Changing both quantities thus helps in mitigating the redundancy problem. More fundamental solutions are suggested but need further research before being effectively implemented.

REFERENCES

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

[2] C. J. Alpert. The ISPD circuit benchmark suite. In *Proc. of the 1998 Intl. Symp. on Physical Design*. ACM Press, 1998.

[3] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: A survey. *Integration: the VLSI Journal*, vol. 19 (no. 1–2): pages 1–81, 1995.

[4] K. Iwama, K. Hino, H. Kurokawa, and S. Sawada. Random benchmark circuits with controlled attributes. In *Proc. of the Electronic Design & Test Conf. on CD-ROM*. ACM, 1997.

[5] 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. of the Design, Automation end Test in Europe Conf.*, pages 656–663. IEEE Computer Society, February 1998.

[6] M. Hutton, J. Rose, and D. Corneil. Generation of synthetic sequential benchmark circuits. In *ACM/SIGDA Intl. Symp. on Field Programmable Gate Arrays*, pages 149–155, 1997.

[7] B. S. Landman and R. L. Russo. On a pin versus block relationship for partitions of logic graphs. *IEEE Trans. on Comput.*, vol. C–20: pages 1469–1479, 1971.

[8] J. Darnauer and W.W. Dai. A method for generating random circuits and its application to routability measurement. In *Proc. 1996 ACM/SIGDA Intl. Symp. on Field Programmable Gate Arrays*, pages 66–72, February 1996.

[9] D. Stroobandt. Generating new benchmark designs for evaluation of CAD tools and new computer architectures. Technical Report DG 98-05, University of Ghent, Belgium, ELectronics and Information Systems Department, April 1998. Available at http://www.elis.rug.ac.be/~dstr/dstr.html.

[10] Y.-C. Wei and C.-K. Cheng. Ratio cut partitioning for hierarchical designs. *IEEE Trans. Comput.-Aided Des., Integrated Circuits & Syst.*, vol. 10 (no. 7): pages 911–921, July 1991.

[11] L. Hagen, A. B. Kahng, F. J. Kurdahi, and C. Ramachandran. On the intrinsic Rent parameter and spectra-based partitioning methodologies. *IEEE Trans. on Comput.-Aided Des., Integrated Circuits & Syst.*, vol. 13 (no. 1): pages 27–37, January 1994.

[12] R. L. Russo. On the tradeoff between logic performance and circuit-to-pin ratio for LSI. *IEEE Trans. Comput.*, vol. C–21: pages 147–153, 1972.

[13] H. Van Marck, D. Stroobandt, and J. Van Campenhout. Towards an extension of Rent's rule for describing local variations in interconnection complexity. In S. Bai, J. Fan, and X. Li, editors, *Proc. 4th Intl. Conf. for Young Computer Scientists*, pages 136–141. Peking University Press, 1995.

[14] E. S. Kuh and T. Ohtsuki. Recent advances in VLSI layout. *Proc. of the IEEE*, vol. 78: pages 237–263, 1990.

[15] D. Stroobandt and F. J. Kurdahi. On the characterization of multi-point nets in electronic designs. In M. A. Bayoumi and G. Jullien, editors, *Proc. of the 8th Great Lakes Symposium on VLSI*, pages 344–350. IEEE Computer Society Press, February 1998.

[16] D. Stroobandt. Analytical methods for a priori wire length estimates in computer systems, November 1998. Ph.D. thesis (English translation from the original text in Dutch), University of Ghent, Belgium, Faculty of Applied Sciences. Available at http://www.elis.rug.ac.be/~dstr/dstr.html.