Cell Replication and Redundancy Elimination During Placement for Cycle Time Optimization

Ingmar Neumann* Dominik Stoffel* Hendrik Hartje+ Wolfgang Kunz*

*J.W.G. University Frankfurt a.M.
Department of Computer Science
Electronic Design Automation Group
60054 Frankfurt a.M., Germany

+University of Potsdam
Department of Computer Science
Fault Tolerant Computing Group
14415 Potsdam, Germany

Abstract

This paper presents a new timing driven approach for cell replication tailored to the practical needs of standard cell layout design. Cell replication methods have been studied extensively in the context of generic partitioning problems. However, until now it has remained unclear what practical benefit can be obtained from this concept in a realistic environment for timing driven layout synthesis. Therefore, this paper presents a timing driven cell replication procedure, demonstrates its incorporation into a standard cell placement and routing tool and examines its benefit on the final circuit performance in comparison with conventional gate or transistor sizing techniques. Furthermore, we demonstrate that cell replication can deteriorate the stuck-at fault testability of circuits and show that stuck-at redundancy elimination must be integrated into the placement procedure. Experimental results demonstrate the usefulness of the proposed methodology and suggest that cell replication should be an integral part of the physical design flow complementing traditional gate sizing techniques.

1. Introduction

With the advent of deep-submicron technologies, interconnect delay has become a dominating factor for circuit delay, thus, dramatically increasing the importance of delay-driven optimization methods at the layout-level.

A method for optimizing circuit speed, being widely used in today's physical design tools, is *gate sizing* or *transistor sizing*, e.g. [1]. Gates or transistors located on critical paths are enlarged to increase their driving force.

A totally different approach for optimizing circuit performance is *cell replication*. Instead of resizing the gates on critical paths, they are replicated so that their fanout tree can be splitted. This allows to concentrate the driving force of a gate onto the critical path. Additionally, the length of the net driven by this gate can often be reduced. Therefore, in many cases the resulting performance gain is higher than if this gate was replaced by an instance of double size.

While gate sizing methods are standard elements in state-of-the-art physical design tools, cell replication strategies have only been investigated in the foreground of generic partitioning problems. However, so far no research has been reported that integrates cell replication into a placement algorithm taking into account the technical requirements to optimize circuit speed during layout synthesis. For example, testability is an important optimization criterion in logic level synthesis. Our research shows that cell replication can drastically reduce the *stuck-at testability* of circuits. This aspect has been completely ignored in previous cell replication literature. Our results show that layout synthesis exploiting the cell replication concept should always be combined with a method to ensure 100% stuck-at testability such as redundancy elimination based on automatic test pattern generation (ATPG).

2. Cell replication for timing-driven placement

2.1 Requirements for Cell Replication Strategies in Circuit Placement

A number of powerful cell replication algorithms have been presented in the past [2] - [7]. While these algorithms have been examined in the context of generic bipartioning problems it is necessary to take into account several practical requirements when cell replication is integrated into a timing-driven placement tool that is based on a recursive procedure performing bipartitioning steps multiple times.

1) The algorithm must be able to handle *multi-terminal nets*, i.e., it must be able to run on a hypergraph representation of the circuit.
2) The algorithm must support *multi-source nets* to handle circuits with bidirectional I/Os, tri-state elements, busses etc..
3) The algorithm must accept size constraints for the area overhead caused by the replication of cells. Cell

0-7803-5832-X /99/$10.00 ©1999 IEEE.

replication is particularly important to redesign small areas of the circuit containing timing critical paths. Therefore, it is crucial that the algorithm can operate with relatively small size constraints so that circuit modifications are focused to small but well-selected regions of the circuit.

4) The algorithm must take into account that the timing data of signal nets can change with every replication of a single cell. Therefore, it is not wise to perform many cell replications at a time, since the cost function is unable to predict the actual timing situation after complex replications.
5) The algorithm must not destroy the stuck-at testability of the circuit that has been achieved by the efforts of logic synthesis.

We briefly review important contributions on cell replication and examine their ability to fulfill the above requirements:

Kring and Newton [2] described an extension to the well known Fiduccia-Mattheyses (FM) algorithm [3] where single cells can not only be moved but also be replicated to minimize the cut size. Since this FM-based strategy only considers single cells, the method is hard to adapt to handle multi-source nets.

In [4][5] Hwang and ElGamal propose to map a given partition of a graph onto a flow network and to apply a min-cut-max-flow algorithm. This method finds an optimal replication set with regard to the given initial partition, but the ignorance of size constraints can lead to very large replication sets. Hypergraphs can be handled by mapping hyperedges onto trees, where the signal source becomes the root node and signal sinks become the leaves. If size constraints are given, the min-cut-max-flow algorithm is replaced by a heuristic based on a directed FM algorithm. Note that either method can only be used in the case, where each signal has exactly one source.

A powerful approach was presented by Liu et al. in [6], where the concept of a *replication graph* is introduced. The algorithm finds the optimal cut size replication in polynomial time. Size contraints are not regarded. In the case of hypergraphs or when size constraints are given a directed FM algorithm is used for partitioning. We have implemented the FM-based version of this approach and conducted several experiments. It turned out that although this method is very powerful for bi-partitioning its application in circuit placement is problematic. To achieve good partitioning results very large size constraints were needed. This leads to large area overheads which is especially unacceptable in a placement procedure where such partitioning is conducted numerous times. Furthermore, as explained above, replications involving a large number of gates are difficult to be guided by a timing-based cost function.

Therefore, we have taken a pragmatic approach and developed a heuristic procedure to integrate cell replication into placement algorithms that attempts to fulfill all the above requirements. This will be described in the next sections.

2.2 The Approach

A well-established method for timing-driven placement is to assign net weights [8][9], which are calculated from the results of a timing analysis and net length estimations. Because the replication of a cell may change the lengths and the slack values of the nets connected to this cell dramatically, we wish to recalculate the weights of these nets immediately after the cell has been replicated. Therefore, accurate timing information can only be obtained if partitioning and replication is carried out in *two separate steps*.

The algorithm is based on recursively partitioning the circuit and chip area. For this purpose the circuit is represented as a weighted directed hypergraph. In each recursion of the partitioning procedure the edge weights of the hypergraph are recalculated for each region based on the minimum reachable net lengths and the upper bounds for the net lengths. The minimum reachable net lengths are estimated immediately before each bipartitioning by using the semi perimeter of the bounding rectangle-method. In this way, the net weights reflect the timing constraints of the circuit. For the calculation of the upper bounds for each signal net the longest path leading through this net is analyzed. This guarantees that for every net an upper bound is calculated. For bipartitioning a region we combined a modified FM-algorithm and a clustering method. Similar to [10], wires leading outside the region of interest are connected to additional dummy nodes being marked as fixed prior to the partitioning step. These nodes are called *position nodes* in this paper and are assumed to be signal sinks. Then, bipartitioning is improved by an optimization step using *cell replication*. The algorithm is able to duplicate single cells as well as sets of cells. When selecting cells for duplication the net weights are evaluated so that preference is given to cells driving nets on the critical paths. After duplicating a single cell or a set of cells, the weights of the nets corresponding to the replicated cells are updated.

Redundancies introduced by cell replication are removed prior to row generation. Testability and replication will be further discussed in section 2.5. Fig. 1 gives an overview of the general program flow.

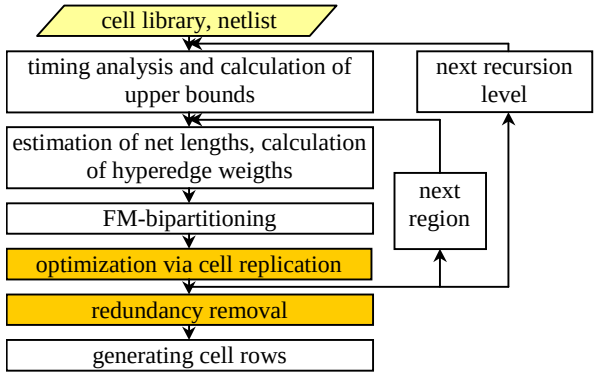

Fig. 1: Placement algorithm

2.3. Bipartitioning and Replication

A key problem when integrating cell replication into a timing-driven placement method is the choice of an appropriate cost function. Therefore, we now describe in more detail the modeling of the circuit and the calculation of the cost function to select cells for replication.

For the partitioning process, a circuit is mapped onto a hypergraph $G=(V, E)$. The node set V consists of three types of nodes:

- *Cell nodes* model the cells of the circuit. The weight of a cell node denotes the area requirement of the cell.
- *Port nodes* model the primary ports of the circuit. During the partitioning process a port node has to stay in a region adjacent to one of the four borders of the circuit area. To ensure this, port nodes can be marked as fixed prior to partitioning. Port nodes have the weight zero.
- *Position nodes* are temporarily introduced to model nets leading outside a region. Position nodes are connected to hyperedges as sink nodes and have the weight zero.

A hyperedge $e \in E$ represents a signal net of the circuit. The connections between the nodes and the hyperedges are provided with a direction to model the signal flow.

A node v is called a source node, a sink node or a bidirectional node with regard to a particular incident edge e depending on the type of the gate port (input, output, bidirectional port) connected to the corresponding signal net.

After the bipartitioning process, which is carried out by a modified FM-algorithm, it is often possible to reduce the cut size if cells are replicated. For illustration Fig. 2 shows a part of a circuit which is divided into two subcircuits located in two subregions R_A and R_B. The cut set contains three signal nets. If the highlighted cell in R_A is replicated so that the copy is placed in R_B, one signal net can be removed from the cut set. In practice this can often simplify the routing task for this net. This is especially advantageous if this net is part of a critical path. In R_B a new signal net is created with the same logic function as the original net in R_A.

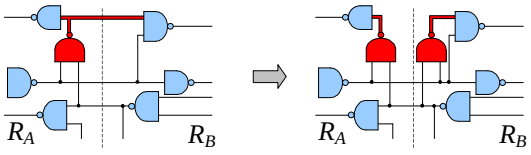

Fig. 2: Cut size improvement via cell replication

In principle, each signal net can be removed from the cut set by replicating its drivers but it should be noted that this may cause other nets to become members of the cut set. This has to be considered in the cost function. If cells contain only one single output port and signals are driven only by one single source, the calculation of the cut size improvement by a particular cell replication is simple. However, as already mentioned, multi-source nets and bidirectional ports must be considered when calculating the cut size gain. In the case of timing driven placement, net weights are calculated on the basis of timing data, which depend on the load driven by the cells. Because these loads change, sometimes dramatically, when cells are duplicated, weights are updated during replication.

Before the replication process is described, some definitions and notations are introduced. Consider a subregion R containing a subgraph of the initial hypergraph:

- $E(v_i)$ denotes the set of hyperedges incident to a node $v_i \in R$.
- $E^o(v_i)$ contains those hyperedges $e \in E(v_i)$, for which v_i is a source node.
- $E^i(v_i)$ contains those hyperedges $e \in E(v_i)$, for which v_i is a sink node.
- $E^b(v_i)$ contains those hyperedges $e \in E(v_i)$, for which v_i is a bidirectional node.

For $E(v_i)$ we obtain: $E(v_i) = E^i(v_i) \cup E^o(v_i) \cup E^b(v_i)$ (1)

Analogously, $V(e_i)$, $V^i(e_i)$, $V^o(e_i)$ und $V^b(e_i)$ denote the sets of nodes v, $v \in R$ which are incident to a hyperedge e_i.

$V^*(e_i)$ contains $V(e_i)$ and additionally all nodes incident to previously created copies of e_i, such that:

$$V^*(e_i) = \bigcup_{(e_j = e_i) \vee (e_j = copy(e_i))} V(e_j) \qquad (2)$$

The driver set D_j of an edge set ES_i denotes the set of nodes that are a signal sources of an edge $e \in ES_j$:

$$D(ES) = \bigcup_{e_j \in ES} \left(V^o(e_j) \cup V^b(e_j)\right) \qquad (3)$$

Furthermore we need the notion of a *candidate set*. A candidate set CS is recursively defined as:

i) a set containing exactly one edge being part of the cut set
ii) the union of two candidate sets CS_i and CS_j, such that $D(CS_i) \cap D(CS_i) \neq \varnothing$

Note that this definition of candidate sets permits to adequately consider multi-source nets when calculating a cost function.

Let D_i denote the driver set for given candidate set CS_i. The set of those hyperedges being signal sources for the nodes $v_i \in D_i$ but not contained in the candidate set is denoted by S_i.

$$S_i = \left(\bigcup_{v_j \in D_i} \left(E^i(v_j) \cup E^b(v_j)\right) \right) \setminus CS_i \qquad (4)$$

Obviously, S_i denotes the set of those hyperedges which would join the cut set if the cut edges $e_j \in CS_i$ were removed from the cut set by duplicating the nodes $v_k \in D_i$. The notion of the driver set facilitates multiple cell replications, e.g., for candidate sets containing busses. Note that a bus can only be removed from a cut set by replicating multiple cells at a time.

The sum of the weights of those edges that would be removed from the cut set is denoted by g_i^+:

$$g_i^+ = \sum_{e_j \in CS_i} w(e_j) \qquad (5)$$

Analogously, g_i^- denotes the sum of the weight of those edges which would join the cut set:

$$g_i^- = \sum_{e_j \in S_i : V_A^*(e_j) \neq \emptyset \vee V_B^*(e_j) \neq \emptyset} w(e_j) \quad (6)$$

$V^*_A(e_j)$ and $V^*_B(e_j)$ denote in the above stated terms two subsets of $V^*(e_j)$ which contain those nodes $v \in V^*(e_j)$ that are part of R_A or R_B, respectively. $V^*_A(e_j)$ and $V^*_B(e_j)$ are used instead of $V_A(e_j)$ and $V_B(e_j)$ to take into account that a previously created copy of e_j may already exist. Since e_j and its copy always show the same logic values, it is not necessary to add e_j to the cut. This concept helps to save edges that cannot be identified by a topological analysis alone. The method keeps track of logically equivalent functions that resulted from cell duplication.

The decrease in cut size g_i gained by the duplication of each hyperedge $e \in CS_i$ reflects the timing improvement of the circuit and can be calculated in the following manner:

$$g_i = g_i^+ - g_i^- \quad (7)$$

As already mentioned, it is necessary to replicate each node $v \in D_i$. The sum of the weights of the new nodes is denoted as the cost dco_i of this replication and reflects the resulting area overhead:

$$dco_i = \sum_{v_j \in D_i} w(v_j) \quad (8)$$

If it is desired to exclude port nodes from replication this is easily taken into account by only considering candidate sets that do not have any port nodes in their driver sets.

As long as there exist candidate sets CS_i with $g_i/dco_i > gain_{min}$, the set with the highest value for g_i/dco_i is selected and the replication is carried out. After the replication for a particular candidate set CS_i, the weights of all nets incident to at least one node v_i, $v_i \in CS_i$, have to be recalculated. A threshold value $gain_{min}$ is used to restrict the replication process to highly weighted critical nets. Note that the construction of this cost function reflects the need to duplicate cells only very selectively to improve local timing values.

The algorithm is described in Fig. 3 in a pseudo-code notation.

```
procedure improve_replication()
repeat {
    Mcs =: set of all candidate sets consisting of single cut edges;
    repeat {
        Mcs^new =: ∅ ;  /* temporary list of new candidate sets */
        for (each CSi, CSj ∈ Mcs with CSi ≠ CSj)
            if (D(CSi) ∩ D(CSj) ≠ ∅ )
                Mcs^new =: Mcs^new ∪ {CSi ∪ CSj};
                Mcs=:Mcs∪Mcs^new
    } while (new candidate set found);
    find candidate set CSi, CSi∈Mcs with maximum value of gi/dcoi;
    if (gi/dcoi > gainmin) {
        carry out replication;
        update w(ei) with ei∈E(vj) with vj∈CSi; }
  } while (at least one replication carried out in this stage);
```

Fig. 3: Replication algorithm

The cost function used is able to handle multi-source nets, e.g., busses, and cells with bidirectional terminals. Timing data and constraints are mapped onto net weights. The changes in timing data caused by cell replication are taken into account by updating the net weights immediately after replicating a candidate set.

2.4 Replication example

Consider the situation shown in the left part of Fig. 4. Since the cut set contains three hyperedges there are three candidate sets with one cut edge. Additionally, there is one candidate set with two cut edges, because e_0 and e_2 have at least one source node in common (v_6). These candidate sets, denoted by $CS_0 ... CS_3$, are described in Table 1. For the calculations of gain and cost unit weights are assumed for nodes and hyperedges. Because CS_3 shows the highest gain-cost relation, the nodes $v \in CS_3$ are duplicated. The reshaped hypergraph is shown in the right part of Fig. 4.

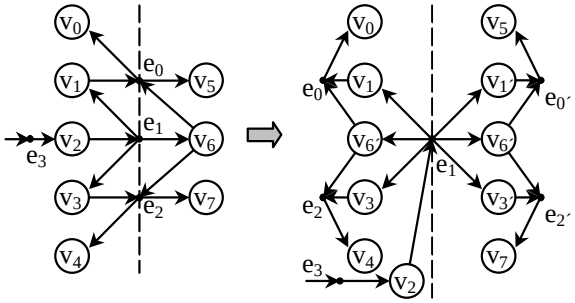

Fig. 4: Replication example

i	CS_i	D_i	S_i	g_i	dco_i	g_i/dco_i
0	e_0	v_1, v_6	e_1	1	2	½
1	e_1	V_2	e_3	0	1	0
2	e_2	v_3, v_6	e_1	1	2	½
3	e_0, e_2	v_1, v_3, v_6	e_1	2	3	2/3

Table 1: Description of the candidate sets of the example from Fig. 4

2.5 Cell Replication and Stuck-At Testability

An important aspect, when looking at cell replication is that the replication of logic cells may lead to untestable stuck-at faults even if the design was 100% stuck-at testable before placement. This invalidates the efforts of synthesis and test preparation and is therefore highly undesirable [11]. Untestable faults can invalidate tests for testable faults and lead to higher costs for test generation and fault simulation.

Fig. 5 shows an example how cell replication causes untestable faults in an initially fully testable circuit.

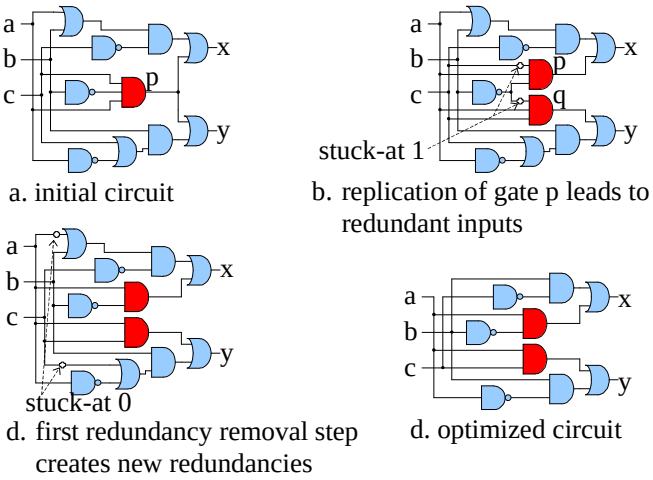

Fig. 5: Redundancy removal

Note that untestable stuck-at faults are always related to circuit redundancy. The removal of this redundancy does not only restore the testability property of the circuit, since wire segments and transistors are removed, it also leads to a more compact solution. In those cases, where the critical path is affected, in general circuit speed is improved, too. The two reasons for this are that short wires have lower capacitance values than long wires, and that gates with a large number of inputs in general have a larger propagation delay than gates with a small number of inputs. For the redundancy removal step a standard ATPG-based approach[11] is used.

3. Experimental results

The replication routine has been implemented as a part of a new timing driven placement and routing package for standard cell designs. The used benchmark circuits have been taken from a MCNC benchmark set and have been mapped onto a standard cell library for a 0.35µm CMOS process with three metal layers. A redundancy check and removal step has been applied to all circuits prior to the experiments. The properties of the used circuits after redundancy removal are shown in Table 2.

Circuit	#cells	#nets	#ports
C432	188	224	43
C499	538	579	73
C880	372	432	86
C1355	642	683	73
C2670	863	1096	372
C3540	1416	1466	72
C5315	2211	2389	301
C6288	2400	2432	64
C7552	3273	3480	315

Table 2: Properties of used benchmark circuits

As explained earlier, the main purpose of this paper is to explore the usefulness of the cell replication concept in the context of a typical timing-driven standard cell placement and routing framework. In order to put the benefits of cell replication into perspective, a thorough comparison with gate sizing techniques is required.

Therefore, we also implemented a gate sizing algorithm which was allowed to introduce the same amount of cell area as produced by the replication approach. In this way we ensure that we compare the performance of two designs that have the same area.

The experimental results are shown in Table 3. The first column contains the circuit name. The following columns contain the area requirement and the minimal cycle time for different design methods. Column 2 and 3 contain the results for timing driven placement without cell replication or gate sizing. Column 4 to 7 show the results using the replication algorithm described in this paper, with and without redundancy removal during placement. The results shown in column 8 and 9 have been obtained by applying the above mentioned gate sizing procedure in the same timing driven placement environment without replication of cells.

circuit	no replication/ gate sizing		replication				gate sizing	
			no red. rem.		with red. rem.			
	A/mm^2	t_{min}/ns	A/mm^2	t_{min}/ns	A/mm^2	t_{min}/ns	A/mm^2	t_{min}/ns
C432	0.0170	5.89	0.0178	5.02	0.176	5.01	0.0181	5.23
C499	0.0360	4.88	0.0395	4.70	0.0395	4.70	0.0394	4.77
C880	0.0307	4.64	0.0327	4.21	0.0320	4.12	0.0327	4.55
C1355	0.0544	5.88	0.0584	5.80	0.0564	5.70	0.0583	5.82
C2670	0.0874	5.06	0.0852	4.28	0.0852	4.28	0.0855	4.42
C3540	0.233	10.0	0.239	9.53	0.234	9.46	0.240	9.92
C5315	0.302	9.51	0.310	8.55	0.307	8.54	0.318	8.90
C6288	0.396	31.3	0.413	29.9	0.401	29.4	0.409	30.9
C7552	0.480	8.91	0.491	8.47	0.490	8.47	0.493	8.62

Table 3: Experimental results

For the used benchmark circuits, a performance improvement of up to 15 % for the replication routine was achieved against the values obtained without replication. Compared to the results obtained by using gate sizing instead of cell replication there is still a speedup of up to 10%. Because the consumed cell area is the same in both cases, our results show that cell replication can make more efficient use of the additionally introduced cell area. It should be noted however, that in both cases, the results can be improved by further gate sizing. The overall optimization potential of gate sizing still is higher than that of cell replication, because every cell can be resized, while only a few cells are suitable for replication. On the other hand, the results show that if there is a potential to improve performance by replicating cells, it should be exploited. Finally, it should be mentioned that in many cases the replication routine also produces somewhat more compact solutions. This can be explained by the simplified routing task for the critical nets.

In the majority of the testcases the replication algorithm created new redundancies. Removing these resulted in

small additional area savings and, in some cases, in further performance improvement.

Table 4 shows the stuck-at fault coverage for the circuits after placement with replication if no redundancy removal is performed.

circuit	#faults	#untestable faults	fault coverage
C432	575	7	98.8%
C499	1287	0	100%
C880	993	1	99.9%
C1355	1814	67	96.3%
C2670	1898	0	100%
C3540	3182	30	99.1%
C5315	5270	14	99.7%
C6288	8844	299	96.6%
C7552	6696	10	99.9%

Table 4: Fault coverage using replication without redundancy removel

It is well-known that very high fault coverages are needed to achieve appropriate *defect coverage* and coverage of non-target faults [11]. Therefore hundreds or even tens of untestable faults are unacceptable in most practical situations. Therefore, it should be noted that for testability reasons alone redundancy elimination must be an integral part in any tool for layout synthesis that performs cell replications.

4. Conclusion

A new approach for optimizing circuit performance using cell replication during placement has been proposed. The replication routine has been integrated into a placement algorithm which is based on recursive partitioning. We used a cost function for cell replication, which is able to handle multi-source nets, e. g., busses, and cells with bidirectional terminals. Timing data and constraints are mapped onto net weights. The changes of timing data caused by cell replication are taken into account by updating the net weights immediately after replicating a candidate set. In comparison with results obtained by a gate sizing approach, it has been shown that the additional cell area introduced by cell replication leads to higher performance improvements than what can be accomplished by the same area overhead for gate sizing.

Cell replication involves the possibility that redundancies are introduced to an initially redundancy-free circuit and stuck-at faults become undetectable. Redundancy removal does not only lead to area savings but also restores the testability property of the circuit. It must therefore be integrated into any practical design flow exploiting the concept of cell replication.

References

[1] Fishburn, J. and Dunlop, A., TILOS: A polynomial programming approach to transistor sizing, Proc. ICCAD-85 (1985) pp. 326-328

[2] Kring, C and Newton, A. R., A Cell-Replicating Approach to Mincut-Based Circuit Partitioning, Proc. ICCAD-91(1991) pp. 2-5

[3] Fiduccia, C and M., Mattheyses, R. M., A linear-time heuristic for improving network partitions, Proc. 19th Design Automation Conference (1982) pp. 175-181

[4] Hwang, L.J. and El Gamal, A., Optimal replication for Min-Cut Partitioning, Proc. ICCAD-92 (1992) pp. 432-435

[5] Hwang, L.J. and El Gamal, A, Min-Cut Replication in Partitioned Networks, IEEE Transactions on Computer Aided Design, vol. 14, no. 1 (1995) pp. 96-106

[6] Liu, L.-T. et al., A Replication Cut for Two-Way Partitioning, IEEE Transactions on Computer-Aided Design, vol.14, no. 5 (1995) pp. 623-629

[7] Mak, W.-K and Wong, D.F., Minimum replication Min-Cut Partitioning, Proc. ICCAD-96 (1996) pp. 205-210

[8] Burstein, M. and Youssef, M. Timing Influenced Layout Design, Proc. 22nd Design Automation Conference (1985), pp. 124-130

[9] Gao, T., A Performance Driven Macro-Cell Placement Algorithm, Proc. 29th Design Automation Conference (1992), pp. 147-152

[10] Dunlop, A. and Kernighan, B., A Procedure for Placement of Standard Cell VLSI Circuits, IEEE Transactions on Computer-Aided Design, vol. 4, no. 1, pp. 92-98, 1985

[11] Abramovici, M. et al., Digital Systems Testing and Testable Design, Computer Science Press (1990)