Design Driven Partitioning

Dirk Behrens, Erich Barke

University of Hanover
Institute of Microelectronic Systems
30167 Hanover, Germany
Tel.: +49 511 762-4986
Fax: +49 511 762-4994
Email: d.behrens@ieee.org

Robert Tolkiehn*

Siemens AG
Semiconductor Devision
80377 Munich, Germany
Tel: +49 89 4144-4970
Fax: +49 89 4144-8983
Email: tolkiehn@hl.siemens.de

Abstract - A new approach for partitioning VLSI digital integrated circuits is presented. In contrast to known approaches, which use only topological information, the presented method also exploits specific information about design modules and higher level design structure. Based on this knowledge, the design driven procedure creates a cluster structure that incorporates the inherent design relationships (e.g. signal flow, logic blocks) in the best way possible. Followed by standard iterative improvement algorithms partitions are produced that outperform many partitioning approaches published before. Because of its linear time complexity the presented clustering strategy is able to handle very large designs. Due to its modular structure it can be easily extended to incorporate special design features or target architectures such as emulation systems.

1 INTRODUCTION

Driven by growing design complexity and new design technologies like MCMs, FPGAs or logic emulation the problem of partitioning digital integrated circuits has regained a high level of attraction during the last five years. Objectives for the partitioning problem are different: In case of MCMs a given circuit has to be split into a given number of parts while minimizing the interconnects between these parts and balancing part sizes [San89]. For design verification by logic emulation the number of parts should be minimal and an upper boundary of part sizes and interconnections is given by device capacity [ChLi94]. When prototyping or implementing a design with multiple heterogeneous FPGAs costs have to be minimized by choosing a cost minimum device distribution [KuBr94].

There are several other goals which normally will be added to the mentioned objectives: First, timing of the design should be optimized. Some approaches do this by critical path analysis [BrSa94]. Second, assignment of feedback loops over more than one part should be avoided. This can be done by an adapted move criterion in iterative improvement algorithms [CoLi94]. Especially for MCMs partitioning for low power has been proposed [KhMa95]. Some other approaches address optimization of a given partition during or following the partitioning process through encoding interconnect information [BaSa93], retiming [LiSh93] or logic replication [HwGa95], [KrNe91].

2 PARTITIONING APPROACHES

Partitioning approaches are classified by the number of parts they produce into bi- and k-way-partitioning algorithms. Most known approaches for k-way partitioning are generalized or modified bi-partitioning algorithms. Therefore, only bi-partitioning is discussed here. Although, many approaches consider only graph partitioning they can be used to partition circuits when applying a clique graph model.

Iterative improvement algorithms, also known as refinement or top-down algorithms, take a (e.g. randomly generated) startpartition and optimize the objective function by interchanging elements [KeLi69] or moving elements [FiMa82]. The quality of partitioning results strongly dependents on the initial startpartition and the random choice of equal-gain nodes for moving. Some approaches overcome this dependency, but then runtime dependents on initial quality. On the other hand, iterative improvement algorithms can easily be adapted to complex objectives like the ones mentioned above.

* This work was done while the author was a graduate student at the University of Hanover, Institute of Microelectronic Systems.

ASP-DAC '97
0-89791-851-7$5.00

Based on Ford & Fulkersons max-flow min-cut theorem [FoFu62] some algorithms use repeated max-flow min-cut [YaWo94], single [Pla90] or multi commodity [LeRa88] flow techniques for partitioning. Network flow approaches produce low cutsizes but on the other hand very unbalanced part sizes.

Recently, algorithms based on quadratic programming techniques, also referred to as spectral or eigenvalue based methods, have been presented showing good partitioning results [RiDo94], [HaKa91]. The general idea of spectral methods is to place the modules into an one-dimensional (using one eigenvalue) or n-dimensional (two or more eigenvalues) space and group neighbored elements into one part.

In contrast to the iterative improvement algorithms there are many approaches based on clustering, also known as bottom-up or constructive algorithms, which try to recursively collapse elements together controlled by some objective function. Usually, these clustering algorithms are used as preprocessing steps for two way iterative improvement algorithms to reduce complexity, runtime of the algorithm and to improve partitioning results. In two-way approaches the iterative improvement algorithm first runs on top of the clustered circuit and uses the results as the startpartition for a second run over the decomposed cluster structure.

The Design Driven Partitioning approach presented in this paper is based on clustering followed by iterative improvement techniques. Therefore, an analysis of known clustering approaches will be given in the next chapter.

3 CLUSTERING ALGORITHMS

Many clustering approaches have been proposed. Generally speaking, it is not possible to differentiate between k-way partitioning (k >> 2) and clustering. Often partitioning algorithms are classified as top-down and clustering algorithms as bottom-up approaches. Clustering algorithms are able to significantly decrease complexity of the problem, i.e. runtime and memory requirements, as well as increase result quality. Many known clustering algorithms try to construct the natural hierarchy of a circuit by finding (strongly) connected components in a design.

Previous work can been grouped into approaches using local or global connectivity information. Simple pairwise local clustering has been presented based on conjunctivity and disjunctivity [ScUl72], average degree [BuHe83], kl-edge connectivity [GaPu90] and time cycles [ShKu92]. Other approaches use cluster radius and sparsity [AwPe90], a compaction heuristic [BuHe89], Rent's Rule [DiHo93], a recursive ratio cut approach [WeCh90], labeling with delay information [MuBr91], random walks [HaKa92], compaction of moved nodes during an iterative improvement algorithm [Saab93], recursive collapsing of small cliques in a graph [CoSm93] and different vertex ordering [AlKa94]. All these approaches use only topological information about a design, except [ShKu92] and [MuBr91] who incorporate timing information.

4 CLUSTERING METRICS

In conjunction with clustering approaches many different clustering metrics exist, which will be used during or following clustering to compare results. All metrics normally apply a weight to each cluster, summarize all weights and scale it with a scale factor. Before explaining our design driven approach in Chapter 5 and 6 we will review the most important cluster metrics here. We use some of these metrics to measure the quality of our clustering approach.

A netlist (hyper)graph $G=(V,E)$ consists of a set of n modules or vertices $V=\{v_1,v_2,...,v_n\}$ and a set of m nets or (hyper)edges $E=\{e_1,e_2,...,e_m\}$. A weight can be assigned to each vertex and each (hyper)edge with some weighting function $v_i \rightarrow w(v_i)$ and $e_i \rightarrow w(e_i)$. A cluster C_i represents a non-empty subset of V and a cluster structure $CS=\{C_1,C_2,...C_k\}$ represents a set of k non-overlapping clusters. Note that contrary to other approaches not every module $v_i \in V$ has to be a member of a cluster C_j.

$$DS: \qquad \max\ f(CS)=\frac{1}{n}\sum_{i=1}^{k}|C_i|\frac{\text{degree}(C_i)}{\text{separation}(C_i)} \qquad (1)$$

The Degree Separation (DS) value [CoHa91] is defined as the average number of nets incident to each module of the cluster, which have at least two pins in the cluster (degree), divided by the average length of a shortest path between any two modules in that cluster (separation).

$$SC: \ \min\ f(CS)=\frac{1}{n(k-1)}\sum_{i=1}^{k}\frac{\left|\{e|\exists u,v\in e,u\in C_i,v\notin C_i\}\right|}{|C_i|} \qquad (2)$$

The Scaled Cost (SC) value [ChSc93] is defined as the weighted sum of cluster outdegree over all clusters divided by cluster size and represents a generalized ratio cut objective.

$$AB: \qquad \max\ f(CS)=\sum_{i=1}^{k}\sum_{\{e\in E|e\cap C_i\neq\emptyset\}}\frac{|e\cap C_i|-1}{|e|-1} \qquad (3)$$

The absorption (AB) metric [SuSe93] adds to each net of a cluster an absorption value between 0 and 1 which implies how much this net is "absorbed" by the cluster.

$$CD: \qquad \max\ f(CS)=\frac{1}{k}\sum_{i=1}^{k}\frac{\sum_{e\in C_i}w(e)}{\binom{n}{2}_{n\in C_i}} \qquad (4)$$

The Cluster Density (CD) [CoSm93] is defined as the total weight of edges in the cluster divided by the number of edges of a complete graph between all modules in the cluster.

$$KL: \qquad max \quad f(CS) = \frac{1}{k}\sum_{i=1}^{k}\frac{k_{C_i}}{l_{C_i}} \qquad (5)$$

Two nodes are kl-connected [GaPr90] if there exist k edge-disjoint paths with length at most l between them. Normally k and l are predefined and a search strategy finds feasible nodes grouped into one cluster. The inherent objective is to find clusters with many edge-disjoint paths of minimum length, which means maximizing k_{C_i} and minimizing l_{C_i}.

5 Design driven clustering

A review of published partitioning and clustering algorithms shows that most of them use topological information about a design exclusively. This is caused by the common partitioning flow, where first a (hyper)graph is created from a flat netlist of the design. Due to this step all design information like hierarchy, "celltypes" of nodes or clocked nets and other useful information is lost. Only the number of nets is used to weight graph edges, sometimes employing timing information or complex graph-weighting-models [HaMa92]. Although nodes could be individually weighted, in most cases the weight is set to one. But information about modules and nets are the most important aspects designers use to manually partition a design and thereby getting better results than any automated algorithm!

The following design information is useful for partitioning: Design hierarchy, sequential and combinational parts, signal directions, feedback loops, busses, large nets like power-nets or global clock-nets, combinational cones, critical paths and most important high level structures like registers or counters which are often difficult to find in a flat design.

In the next chapter we show how to use this design information to build up a cluster structure, which we call "design driven cluster structure". Following we give an overview of the clustering process explaining all steps in detail. In Chapter 6 we analyze the final cluster structure using the metrics mentioned above. Finally (Chapter 7) we present partitioning results from a standard iterative improvement algorithm running on top of the cluster structure finishing in Chapter 8 with a short survey of possible extensions for the presented approach.

Our clustering flow consists of five different phases:

1) Calculate non-overlapping cluster structures (CS) using different design information. Here we use cones, connected components and feedback loops.
2) Merge cluster structures to one non-overlapping structure
3) Rating the different clusters and eliminating "bad" clusters
4) Grow clusters with free elements
5) Rating the final cluster structure and eliminating "bad" clusters

Fig. 1 and Fig. 2 show some statistics about the overall clustering process for the MCNC benchmark circuit primary1. These are typical statistics which are very similar for other circuits. Following this flow overview we will explain all steps in more detail:

Cone clustering: There are several approaches using cones for clustering, showing good results [BrSa94], [CoLi94]. Mainly, cones are used to avoid cutting of critical paths during partitioning. Consequently, our first CS is based on cones. We use fanout free cones to avoid overlapping of different clusters. As to be seen in Figs. 1 and 2 the fanout free cone clustering step clusters only 30% of all instances and produces many small clusters with only 2 or 3 instances. Cone clustering has time complexity $O(d \cdot |V|)$, with d as average number of nodes a node is connected to.

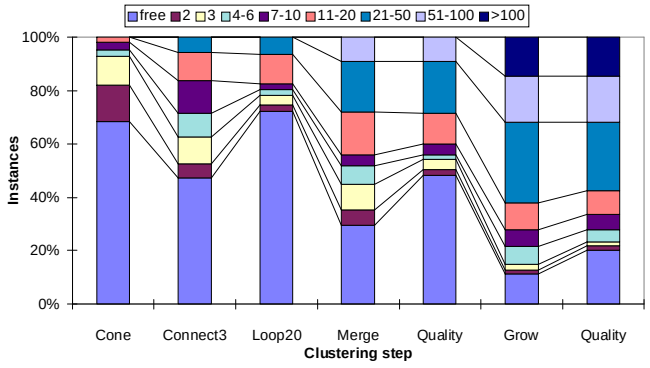

Fig. 1: Grouping of instances into clusters of specific sizes (primary1)

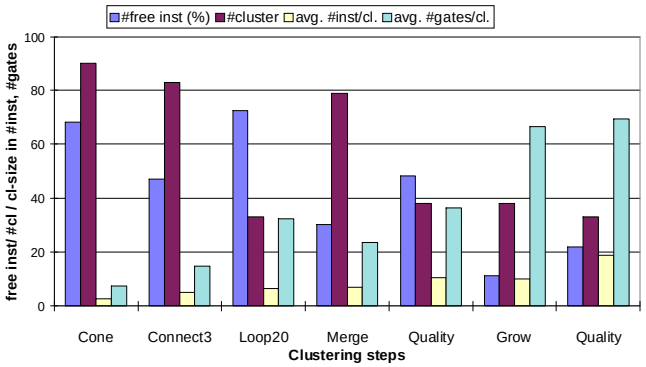

Fig. 2: Number of free instances, number of clusters and cluster sizes in instances and gate-equivalents during clustering process (primary1)

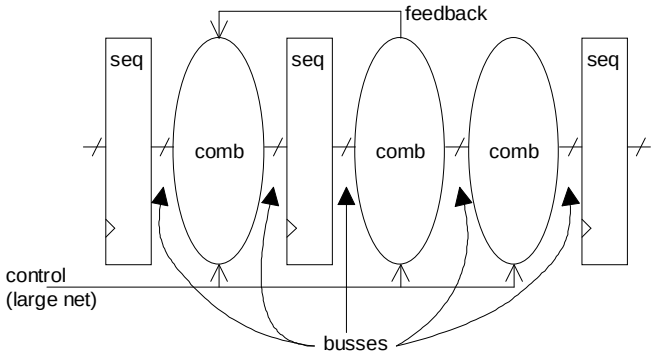

Fig. 3: Sequential and combinational parts in a pipelined design

Connected components: The idea of clustering connected components is to consider structures like pipelined designs, with large combinational parts. To find connected components (Fig. 3), we first remove the clocktree which consists of all clocked elements and combinational elements driving clocked input pins. Then we remove busses. Finally, we remove big nets connecting more than a predefined number of elements and apply a standard connected components algorithm. Figs. 1 and 2 show that if removing all nets with more than three pins (Connect3) this step clusters more than 50% of all instances. The generated clusters are larger (up to 50 instances) than those produced by cone clustering (about 20 instances only). Finding connected components has $O\left[|V|+|E|\right]$ time complexity.

Feedback loops: Considering the timing of a partitioned design, feedback loops should not be cut. Like [CoLi94] we try to keep all feedback loops in one part. Therefore, we use a modified Depth First Search to find all feedback loops in a design. Because in many designs there are very long loops, which cannot be captured in one part, we limit the size by a predefined maximum number of elements in the loop. As Figs. 1 and 2 show, clustering of loops with less or equal 20 elements in the loop (Loop20) produces a small number of clusters with good sizes (11 to 50 elements). Most times several loops are combined within one cluster. The used Depth First Search approach has complexity of $O\left[|V|+|E|\right]$.

Merging: The next step is to merge cluster structures. By merging overlapping clusters we form a new cluster. If the new cluster exceeds given gatecount or pincount limits the cluster of highest priority is preserved and the other ones are decomposed again. Priority of feedback loop clusters is higher than that of cone clusters, which in turn is higher than that of connected components clusters. This trivial merging approach will be improved by future work. As shown in Figs. 1 and 2 more than 30% of all instances are grouped into clusters with 11 - 100 elements by merging cone, connect and loop clusters. Only 30% of all instances are not clustered. However, there still exist some smaller clusters having pincount and netcount to gatecount ratios, which are not desirable for the following partitioning step. These clusters will be recognized and decomposed by the following quality rating step. Merging has time complexity of $O\left[|CS|^2\right]$, which is most times less than $O\left[|V|+|E|\right]$.

Quality rating: For qualifying clusters we can use any of the above clustering metrics. However, some of them address only a subset of the objectives and the more complex ones require a lot of time to evaluate. Therefore, we define a different metric: Its first objective is to get "dense" clusters, which means that it favours a lot of nets captured in each cluster with respect to the number of instances included. Second, it minimizes the number of clusterpins with respect to the number of instances included. This results in the following Cluster Quality (CQ) metric:

$$CQ: \quad \max\ CQ = \frac{1}{k}\sum_{i=1}^{k}\frac{|E|_{e\in C_i} - |E|_{\exists v,w\in e,\, v\notin C_i \wedge w\in C_i}}{|V|_{v_i\in C_i}} \qquad (6)$$

Applying this metric to all built clusters, we got good results by decomposing all clusters with a cluster quality value of less or equal -1. Many small clusters with size between 2 and 10 instances are decomposed and almost all big clusters are preserved. The disadvantage of this quality step is the increasing number of free instances from 30% to 50%. To overcome this problem we let all clusters grow in a following step. By using an intelligent cluster storing, quality rating only has time complexity of $O\left[|CS|\right] << O\left[|V|+|E|\right]$.

Growing: During the loop detection process all feedback loops have already been cut and the complete design has been ranked by a Breadth First Search. We now use this ranking to traverse all free instances, and if one of them is directly connected to a cluster we compare its rank r_i to that of the neighboring element r_n in the cluster. If $r_n \in \{r_i+1; r_i-1\}$ the instance will be added to the cluster. Otherwise we proceed with the next directly connected cluster. If no connected cluster fits the ranking objective we proceed with the next free instance. As in the merging step, clusters that reach given limits in gate or pin count are locked for any further growing process. After the growing procedure approximately 60% of all instances are grouped into clusters with 21 - 100 and more elements. A following second quality rating step will preserve these big clusters and only decompose a few of the smaller ones (Figs. 1 and 2). At the end about 80% of all instances have been clustered with an average cluster size of 18,64 instances per cluster and only 20% of all instances are free (primary1). For some circuits we got improved results by applying the quality and growing steps more than once. Fig. 4 gives a pseudo code listing of the clustering process. The described growing process has time complexity of $O\left[|V|_{free} * |CS|\right]$.

```
// Get cluster structures
foreach  designinformation i=0 to n do
        create non-overlapping cluster structure CS_i

// Merge cluster structures
for i=1 to n do
        merge CS_0 with CS_i

// Eliminate "bad" clusters
forall C_i ∈ CS_0 do
        if ClusterQuality(C_i) < CQ threshold value then
                decompose C_i

// Let cluster grow
forall free instances I_i do
        if I_i has connections to at least one cluster C_j
                if rank(I_i) = rank(I_Cj)±1 then
                        add I_i to Cluster C_j

// Eliminate "bad" clusters
forall C_i ∈ CS_0 do
        if ClusterQuality(C_i) < CQ threshold value then
                decompose C_i
```

Fig. 4: Pseudo code for the presented Design Driven Clustering approach

Fig. 5 shows the evolution of different clustering metrics during the clustering process. Because of different absolute values they are standardized first. Note that the objective is to maximize DS, AB and CQ while minimizing SC respectively -CQ. Cone clustering produces average DS values but poor SC, AB and CQ values.

Connected components clustering results in average DS, SC and AB values. Loop clustering produces good DS values, average SC and CQ and poor AB values. The following merging step has great influence in decreasing the SC value and increasing the AB value. Succeeding quality rating improves only the SC value.

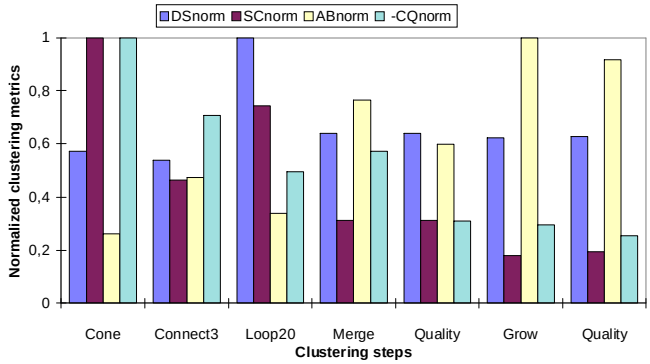

Fig. 5: Evolution of different clustering metrics during the clustering

A global view of the evolution shows increasing AB values and decreasing SC and -CQ values. The DS value is well suited for all clustering aspects although merging and quality rating do not improve it. However, good partitioning results as shown in the next chapters prove the quality of our cluster strategy.

6 Clustering results

For comparing the quality of our clustering approach we have clustered different circuits from the MCNC benchmark suite and two industrial test cases and compared the results to previous publications. An overview of cluster counts and sizes is summarized in Table 1. Reduction of problem size is shown in Table 2. Values of some clustering metrics compared to other published clustering approaches are presented in Table 3.

Table 1
Number of clusters of specific size and DS, SC, AB and CQ values for N=3, L=20

design	instance count								clustering metrics			
	2	3	4-6	7-10	11-20	21-50	51-100	>100	DS	SC	AB	CQ
primary1	6	5	3	5	4	8	1	1	1,13	208,5	563	-0,52
primary2	7	6	7	5	3	9	7	6	0,83	67,1	1503	-0,61
s9234	10	16	58	52	37	18	9	3	0,49	15,2	3141	0,30
bio	0	0	398	46	3	27	1	8	1,05	23,7	4385	-0,55
ind1	7	13	15	5	8	6	1	5	0,65	79,0	1393	-0,64
ind2	26	7	11	9	7	14	10	9	0,72	50,4	2560	-0,66

Table 2
Reduction factors for instances and nets for N=3, L=20

design	instances + clusters		netcount		reduction	
	netlist	clustered	netlist	clustered	instances	nets
primary1	787	183	881	403	77%	54%
primary2	3036	2201	3148	2585	28%	18%
s9234	3358	189	3349	457	94%	86%
bio	5533	601	4865	891	89%	82%
ind1	1703	950	1746	1271	44%	27%
ind2	3618	2381	4523	3162	34%	30%

Table 3
DS, SC, AB values cluster sizes of different approaches from [AlKa94] and [CoSm93]

design	clust.alg.	DS	SC	AB	CD	avg. clst size
primary1	WINDOW	1,471	173,10	687,60		4,36
	RW-ST	1,325	287,90	629,90		4,36
	AGG	0,879	277,90	437,00		4,36
	MBC	1,258	254,00	309,30		4,36
	FMC				0,176	7,40
	DDP	1,130	208,50	562,60	0,150	16,15
primary2	WINDOW	1,539	57,69	2257,00		4,29
	RW-ST	1,566	82,81	2013,00		4,29
	AGG	1,048	89,73	1227,00		4,29
	MBC	1,238	82,44	736,40		4,29
	FMC				0,248	12,10
	DDP	0,830	67,1	1502,72	0,305	37,94

As shown in Table 1, Design Driven Partitioning produces cluster structures with most clusters of size 4-50. However, also large clusters with more than 100 instances will be produced. The cluster structures have reduction factors from 28%-94% for instances and 18%-86% for nets (Table 2). Compared to other sometimes very complex clustering approaches presented in [AlKa94] and [CoSm93] DDP shows good SC, AB, CD and avg. cluster sizes (Table 3).

7 PARTITIONING RESULTS

In this chapter we will demonstrate the quality of our clustering scheme by applying a bi-partitioning procedure on the calculated cluster structure. Therefore, we implemented the standard Fiduccia and Mattheyses algorithm as described in [FiMa82]. With this algorithm we bi-partition the clustered MCNC benchmarks and industrial circuits presented in the Chapter 6 using three runs with different starting partitions for each design. We then compare these results (DDP) with already published results from PARABOLI [RiDo94], FM and FMC [CoSm93], FM and RW-ST [HaKa92], MBC [Bui89], IG-Match [CoHa92], EIG1-IG [HaKa92], EIG1 [HaKa91] and FM and RCut 1.0 [WeCh91]. Note that we have transferred results published under primary1/2, which do not use module size information, to primSC1/2 and primGA1/2 respectively.

As seen in Figs. 6 and 7 a standard FM algorithm produces good results regarding net cut size and ratio cut values. For primary1 regarding cutsize DDP outperforms clustering approaches published before, e.g. MBC, RW-ST and FMC. Eigenvalue based methods, e.g. IG-Match, EIG1-IG and PARABOLI, produce smaller cutsizes but are also outperformed regarding RatioCut values. For primary2 cutsize of DDP is not very impressing. This results from a strong clockpath with many combinational elements in the clocktree, which leads into an inefficient cluster structure (see future work).

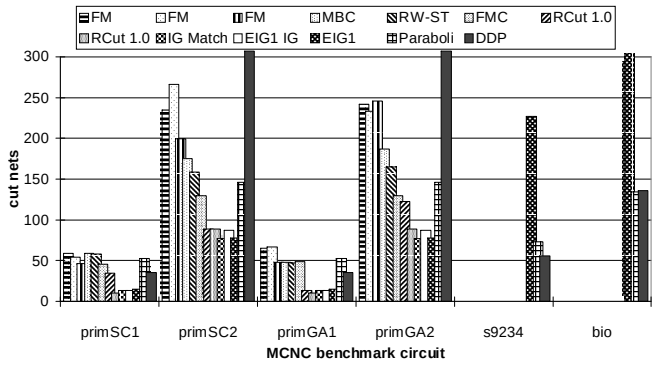

Fig. 6: Cutsize for bi-partitioning of clustered designs with standard FM

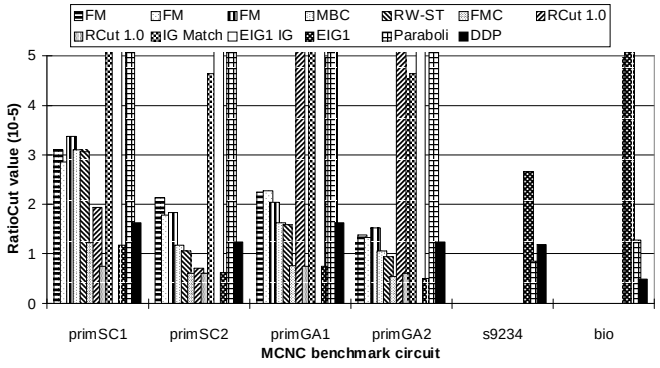

Fig. 7: RatioCut for bi-partitioning of clustered designs with standard FM

But RatioCut value outperforms simple FM and eigenvalue based approaches. For s9234 DDP produces best cutsize ever published and for bio it produces comparable low cutsize and best RatioCut value.

8 CONCLUSION AND FUTURE WORK

As shown by the presented results design driven clustering produces comparable results with low time complexity. Finding higher level logical and structural clusters in a design does not only decrease cutsize but also can offer the opportunity to include other objectives like timing optimization automatically. By using different cluster structures as well as a specialized growing process the presented approach is expandable in several ways. Incorporating additional design information promises higher quality results.

Due to growing design complexity all top down approaches will eventually reach their limits regarding computing time and memory requirements. Therefore, bottom up approaches are favorite candidates to handle upcoming complexities. However, the partitioning results presented show that iterative improvement algorithms like FM are still acceptable approaches if reasonable preprocessing steps to reduce complexity are used.

We currently extend our approach with respect to the cluster structure by incorporating other design information like design hierarchy and critical paths. In preliminary examinations (Table 4) two industrial designs have been partitioned as flat netlists (FM), using the hierarchy as a clustering structure (FM + hierarchy) and using our DDP on the flat design. Results show that using hierarchy information in some cases will drastically reduce cutsize, where DDP today does not show impressive results (ind1), while in other cases using hierarchy will not be very efficient, but DDP is (ind2). But always DDP produces best RatioCut values. So incorporating hierarchy information into the presented clustering strategy promise additional improvement of results.

TABLE 4
BI-PARTITIONING RESULTS FOR STANDARD FM, FM USING HIERARCHY AND DDP APPROACH

design	cutsize			RatioCut		
	FM	CH+FM	DDP	FM	CH+FM	DDP
ind1	141	47	128	8,75E-05	1,72E-04	6,92E-05
ind2	559	515	258	1,95E-05	1,79E-05	8,96E-06

To handle designs with large sequential parts, clustering of the clocktree using ranking information will be added. Thereby, a defined number of large clock nets will be handled as global nets and cut by default. Also, cluster quality rating and cluster growing steps will be enhanced to control growing for specific FPGA demands. Finally we will extend this approach to distribute a given design onto a minimum number of FPGAs by using only the growing process without the need of a following partitioning step.

9 REFERENCES

[AlKa94] Alpert, C.J.; Kahng, A.B.; "A General Framework for Vertex Orderings With Applications to Netlist Clustering", Int. Conf. on Computer Aided Design, pp. 63-67, 1994

[AwPe90] Awerbuch, B.; Peleg, D.; "Sparse Partitions", IEEE Ann. Symp. on Found. of Computer Science, pp. 503-513, 1990

[BrSa94] Brasen D.; Saucier G.; "FPGA Partitioning for Critical Paths", Proc. European Design Automation Conf., pp. 99-103, 1992

[BuHe83] Bui T.; Heigham C.; Jones C.; Leighton T.; "Improving the Performance of the Kernighan-Lin and Simulated Annealing Graph Bisection Algorithms", Proc. Design Automation Conf., pp. 775-778, 1989

[ChLi94] Chou, N.-C.; Liu, L.-T.; Cheng, C.-K.; Dai, W.-J.; Lindelof, R.; "Circuit Partitioning for Huge Logic Emulation Systems", Proc. Design Automation Conf., pp. 244-249, 1994

[ChSc93] Chan P.K.; Schlag M.D.F.; Zien J.Y.; "Spectral K-Way Ratio-Cut Partitioning and Clustering", Proc. Design Automation Conf., pp. 749-745, 1993

[CoHa91] Cong, J.; Hagen, L.; Kahng, A.; "Random Walks for Circuit Clustering", Int. ASIC Conf., pp. P14-2.1 - P14-2.4, 1991

[CoHa92] Cong, J.; Hagen, L.; Kahng, A.; "Net Partitions Yield Better Module Partitions", Proc. Design Automation Conf., pp. 47-52, 1992

[CoLi94] Cong J.; Li Z.; Bagrodia R.; "Acyclic Multi-Way Partitioning of Boolean Networks", Proc. Design Automation Conf., pp. 670-675, 1994

[CoSm93] Cong J.; Smith M.; "A Parallel Bottom-Up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design", Proc. Design Automation Conf., pp. 755-760, 1993

[DiHo93] Ding C.; Ho C.; "A New Optimization Driven Clustering Algorithm for Large Circuts", Proc. European Design Automation Conf., pp. 28-32, 1993

[FiMa82] Fiduccia C.; Mattheyses R.; "A Linear-Time Heuristic for Improving Network Partitions", Proc. Design Automation Conf., pp. 175-181, 1982

[FoFu62] Ford, J. R. and Fulkerson, D. R.; "Flows in Networks", Princeton University Press, 1962

[GaPr90] Garbers, J.; Prvmel, H.J.; Steger, A.; "Finding Clusters in VLSI Circuits", Int. Conf. on Computer Aided Design, pp. 520523, 1990

[HaKa91] Hagen, L.; Kahng, A; "Fast Spectral Methods for Ratio Cut Partitioning and Clustering", Int. Conf. on Computer Aided Design, pp. 10-13, 1991

[HaKa92] Hagen L.; Kahng, A.B.; "New Spectral Methods for Ratio Cut Partitioning and Clustering", IEEE Transactions on Computer Aided Design, vol. 11, no. 9, pp. 1074-1085, 1992

[HaMa92] Hadley S.W.; Mark B.L.; Vannelli A.; "An Efficient Eigenvector Approach for Finding Netlist Partitions", IEEE Transactions on Computer Aided Design, vol. 11, no. 7, pp. 885-892, 1992

[HwGa95] Hwang J.; Gamal, A; "Min-Cut Replication in Partitioned Networks", IEEE Transactions on Computer Aided Design, vol. 14, no. 1, pp. 96-106, 1995

[KeLi69] Kernighan B.; Lin S.; "An Efficient Heuristic Procedure for Partitioning Graphs", The Bell System Technical Journal, pp. 291-307, 1969

[KhMa95] Khan, S. A.; Madisetti, V. K.; "System Partitioning of MCMs for Low Power", IEEE Design & Test of Computers, Spring 1995, pp. 41-52, 1995

[KrNe91] Kring C.; Newton A.; "A Cell-Replicating Approach to Mincut-Based Circuit Partitioning", Int. Conf. on Computer Aided Design, pp. 2-5, 1991

[KuBr94] Kuznar, R.; Brglez, F.; Zjac, B.; "Multi-way Netlist Partitioning into Heterogeneous FPGAs and Minimization of Total Device Cost and Interconnect", Design Automation Conf., pp. 238-243, 1994

[LiSh93] Liu, L.-T.; Shih, M.; Chou, N.-C.; Cheng, C.-K.; Ku, W.; "Performance-Driven Partitioning Using Retiming and Replication", Int. Conf. on Computer Aided Design, pp. 296-299, 1993

[MuBr91] Murgai R.; Brayton R.; Sangiovanni-Vincentelli A.; "On Clustering for Minimum Delay/Area", Proc. Design Automation Conf., pp. 6-8, 1991

[Pla90] Plaisted, D. A.; "A Heuristic Algorithm for Small Separators in Arbitrary Graphs", SIAM Journal on Computing, vol. 19, no. 2, pp. 267-280, 1990

[RiDo94] Riess B.M.; Doll K.; Johannes F.M.; "Partitioning Very Large Circuits Using Analytical Placement Techniques", Proc. Design Automation Conf., pp. 646-651, 1994

[Saab93] Saab Y.; "Post-Analysis Based Clustering Dramatically Improves The Fiduccia-Mattheyses Algorithm", pp. 22-27, 1993

[San89] Sanchis, L. A.; "Multiple-way network partitioning", IEEE Transactions on Computer, vol. 38, no. 1, pp. 62-81, 1989

[ScUl72] Schuler D.; Ulrich E.; "Clustering and Linear Placement", Design Automation Workshop, pp. 50-56, 1972

[ShKu93] Shih, M.; Kuh, E.S.; "Quadratic Boolean Programming for Performance-Driven System Partitioning", Proc. Design Automation Conf., pp. 761-765, 1993

[SuSe93] Sun, W.; Sechen, C.; "Efficient and Effective Placement for Very Large Circuits", Proc. Design Automation Conf., pp. 170-177, 1993

[WeCh90] Wei Y.; Cheng C.; "A Two-Level Two-Way Partitioning Algorithm", Int. Conf. on Computer Aided Design, pp. 516-519, 1990

[WeCh91] Wei Y.; Cheng C.; "Ratio Cut Partioning for Hierarchial Design", IEEE Transactions on Computer Aided Design, vol. 10, no. 7, pp. 911-920, 1991

[YaWo94] Yang, H.; Wong, D.F.; "Efficient Network Flow Based Min-Cut Balanced Partitioning", Int. Conf. on Computer Aided Design, pp. 50-55, 1994