OPTIMIZING LARGE COMBINATIONAL NETWORKS FOR K-LUT BASED FPGA MAPPING

Ion I. BUCUR, *PhD*
Ioana FĂGĂRĂŞAN, *PhD*
Cornel POPESCU, PhD
University Politehnica of Bucharest

George CULEA, PhD
University of Bacău, Faculty of Electrical Engineering,

Alexandru E. ŞUŞU, PhD
Swiss Federal Institute of Technology Lausanne

Abstract: Optimizing by partitioning is a central problem in VLSI design automation, addressing circuit's manufacturability. Circuit partitioning has multiple applications in VLSI design. One of the most common is that of dividing combinational circuits (usually large ones) that will not fit on a single package among a number of packages. Partitioning is of practical importance for k-LUT based FPGA circuit implementation. In this work is presented multilevel a multi-resource partitioning algorithm for partitioning large combinational circuits in order to efficiently use existing and commercially available FPGAs packages.

Keywords: two-way partitioning, multi-way partitioning, recursive partitioning, flat partitioning**,** critical path, cutting cones, bottom-up clusters, top-down min-cut.

1. Introduction

Partitioning is a technique of dividing a circuit or system into a collection of smaller blocks (sub-circuits) with roughly equal sizes targeting to minimize the number of interconnections between the blocks.

It is, on the one hand, a design task to break a large system into pieces to be implemented on separate interacting components and, on the other hand, it serves as an algorithmic method to solve difficult and complex combinatorial optimization problems as in logic or layout synthesis. Partitioning has been an active area of research for at least a quarter of a century [1, 2].

The main reason that partitioning has become a central and sometimes-critical design task today is the enormous increase of system complexity in the past and the expected further advances of deep sub-micron electronic system design and fabrication. Soaring system complexities result from a combination of reasons:

- Increasing circuits complexity and
- Shorter turn-around time to reach the market with new products.

Broadly accepted powerful high-level synthesis tools allow the designers to automatically generate huge systems. In a functional specification, by just changing a few lines of code, the size of the resulting structural description (net list) of a system can increase dramatically.

Synthesis and simulation EDA tools often hardly cope with the complexity of the whole system under design, and engineer aim is to concentrate on critical parts of a system in order to speed-up design cycle. It results that the present state of design technology often requires a partitioning of the system [3, 4, 5].

Fabrication technology makes increasingly smaller feature sizes and augmented die dimensions possible, thus allowing a circuit to accommodate huge number of transistors. However, circuits are restricted in size and in the number of external I/O connections. FPGAs devices are an appropriate example [5, 6].

Fabrication technology, obviously, requires the partitioning of a system into components. Economical pressure yields larger systems, both to make production cheaper and to exploit the optimization potential of the complete system. The various parts of the system should be implemented in appropriate ways to achieve low-cost fabrication, optimal system performance, and easy adaptation to changing requirements. Thus, profit can be received by partitioning a system optimally [6, 7, 8, 9].

Partitioning applications exist on all levels of abstraction, specifically on the functional (behavioral) and on the structural (net list) level. In the early stages of design, far-reaching decisions have to be made how to partition a design, often based on incomplete knowledge.

It has been observed that structure synthesis tools, in general, do not generate a hierarchy that can be used directly for mapping (FPGAs case) or for layout design if this hierarchy is deep [9, 10]. To give the mapping and layout synthesis tools the freedom they require to generate good results; net lists have to be flattened out and repartitioned [9, 10, 11]. In particular, it has to be decided whether to implement a component in various types of hardware technologies to achieve an optimal size/performance trade-off. Because the granularity is low in such situation, i.e. relatively few objects of moderate to high complexities, human designers based on their experience can possibly do partitioning [5, 6, 9].

The components resulting from system partitioning are implemented by a team of designers or synthesized from a high-level description by using synthesis tools that generate a structural implementation [3, 4, 9].

Field Programmable Gate Arrays (FPGAs), providing both large-scale integration and user-programmability, are important circuit architectures. These features have enormous impact on reducing integrated circuit manufacturing time and costs. FPGA packages, as a general feature, have maximum size CLBs constraints much larger than the number of input-output pins IOBs.

Thus, implementation of a large logic network into working FPGA involves network partitioning into a near balanced packing of Combinational Logic Blocks (CLBs) and Input-Output Blocks (IOBs). Resulting IOBs bottleneck during circuit partitioning could involve more required devices and possibly more ordinary signal wires crossing between packages. It implies more critical timing paths between packages and drastically decreases frequency operational of the circuits. Critical paths are long combinational path between sequential elements and IOBs. Cutting critical paths during circuit partitioning into separate packages implies capacitances of packages interconnections that could drastically reduce network speed [6, 9, 10, 11, 12, 13, 14, 15].

FPGA circuit implementation has two main phases. Placement phase, the first one, is dedicated to assign desirable locations within the FPGA structure, to the obtained blocks. This could be, however, an iterative process. Routing phase, the last one, provides the interconnections between these blocks [6, 16, 17, 18, 19, 20, 21, 22]. Circuit partitioning is used, however, twice in FPGA implementation. First usage concerns too large designs to fit available FPGA packages. A less obvious usage of network partitioning is used in the blocks placement phase [21, 22, 23, 24]. Placement algorithms based on circuit partitioning yields astonishing results efficiently.

In this work is presented multilevel multi-resource partitioning algorithm for partitioning large combinational circuits in order to efficiently use existing and commercially available FPGAs packages.

2. Published work

Typical partitioning objectives such as minimum-width bisection and minimum ratio cut are *NP-complete* and require such heuristics as simulated annealing, greedy *k-opt* interchange or quadratic optimization (via relaxation or spectral methods). Hopefully these heuristics are computing fine solution close enough to the optimal one.

The objective of two-way partitioning, [2] is to either minimize the cut-size when partitioning the network into two (roughly) equal-size blocks, or to minimize the ratio cut size between the two blocks [25, 26]. The two-way partitioning algorithms include the Kernighan-Lin [1] successful heuristic and iterative improvement methods [1] the graph spectrum method [17], and the net-based partitioning method [18, 19].

The multi-way partitioning algorithms include the recursive Kernighan-Lin two-way partitioning method, a generalization of the spectrum-based partitioning method, [8], the generalization of the FM-algorithm [16] with look-ahead scheme, [3, 12]. Most recent years a number of new thoughts have been introduced supplementary improving the quality of partitioning solutions, including communication-complexity based partitioning [4], cluster-based partitioning methods [14], and partitioning with module replication [20, 23].

3. Problem formulation

In this paper, it is studied the partitioning problem for combinational Boolean networks. A combinational Boolean network C can be represented as a directed acyclic graph $G = (V, E)$ where each node n ($n \in V$) represents a logic gate and a directed edge (i, j), ($(i, j) \in E$) exists if the output of gate i is an input of gate j. A primary input (PI) node has no incoming edge and a primary output (PO) node has no outgoing edge. A *disjoint Q-way partitioning solution* $S = (A1, A2... AK)$ satisfies the following conditions:

$Ai \cap Aj = \phi$ for $i \neq j$ and

$\cup\, Ai$, $0<i<Q+1$, contains all the gates in the network;

$A1, A2... AK$ are known as *clusters of* G (C).

Each node in C has only one output line and limited number of input lines. It is used *input*(v) to denote the set of fanins of gate. Given a subgraph H of the Boolean network, let *input*(H) denote the set of *distinct* nodes outside H, which supply inputs to the nodes in H (*fanins* of H). For a node n in the network, a *w-feasible cone at n*, denoted K_n, is a subgraph consisting of node n and its predecessors (u is a predecessor of n if there is a directed path from u to n), such that $|input(K_n)| \leq w$ and any path connecting a node in K_n and n lies entirely in K_n.

The *level* of a node is the length of the longest path from any PI node to n. The level of a PI node is zero. The *depth* of a network is the *largest node level* in the network. A Boolean network is *p-bounded* if $|input(n)| \leq p$ for each node n in the network.

Since it is always attractive having disjointed partitioning solutions, the word '*disjoint*' might be omitted in later discussions. The main objective is to minimize the total number of nets between different partitions.

Moreover, for a multi-way partitioning solution S, one can define a directed graph $D(S)$, called the *dependency graph* of S, such that each node in $D(S)$ represents a block in S, and there is a directed edge (Ai, Aj) in $D(S)$ if and only if there exists an edge (x, y) in C such that $x \in Ai$ and $y \in Aj$.

The assumption that it is given a combinational network guarantees the existence of disjoint partitioning solution. When it is given a general net list, one can first remove all the sequential elements in order to obtain only a combinational network, [21, 22, 11]. Most of existing partitioning methods model a network as an undirected graph or hyper graph, and ignore the signal directions during the partitioning process.

However, the study in this paper shows that considering signal directions is very helpful in identifying the underlining circuit structure, which can lead to significant improvement on the partitioning results.

4. Cluster partitioning algorithm

Cluster partitioning algorithm was implemented using SIS-1.2 structures and routines and most of the terminology used in this paper is similar to the terminology used in SIS-1.2 documentation.

Implemented algorithm split-up C using directed acyclic graph G (as model of this combinational Boolean network), before mapping *K-LUT nodes in the circuit.* Combinational circuits could be very large and cluster partitioning helps obtaining more technological compliant mapping over the initial circuit.

Before starting the first network traversal, all nodes are inserted in a partial-ordered structure, such that each node n_i feeding node n_j appears before n_j in this structure. Each internal node structure has an additional array denoted *po_label*, mapping all POs nodes of the circuit; (*po_label*(β) is mapping PO$_\beta$, as an example). This array it's initialized with zero.

First traversal, depth first search from outputs, establish nodes affiliation with respect to the *primary output nodes*. Primary output nodes in Fig. 1 are *z*, *x*, *y*, and *w*. An internal node having more than one element not zero in its *po_label* belongs to more than one primary output transitive cone, and it's said to be *multiple dominated*.

If node *n* belongs to the transitive cones of PO$_1$, PO$_2$ and PO$_3$, as an example, than *po_label*(1) = *po_label*(2) = *po_label*(3) = 1. All such nodes are defining sub-cone(1,2,3) as the intersection of the three mentioned cones.

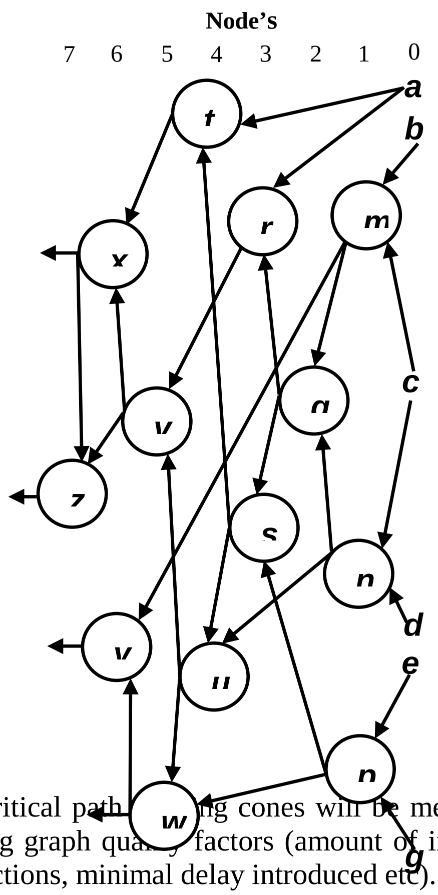
Node *t*, in Fig. 1, has *po_label* marking *w, x, y,* and *z* affiliation, while primary output node *w* has affiliated only node *y*.

k-LUT mapping is made over *homogenous dominated cones*. It means that all nodes dominated only by *z*, or by *z* and *t,* as an example, will be mapped in a separate mapping session. This strategy separates nodes having fan-outs in more than one single output cone and avoid interactions during mappings in different primary output cones.

Mapping phase starts by considering nodes that belongs to the set of critical paths. The primary output node z and all nodes belonging to the transitive cone rooted in this node define critical path, in Fig. 1, as an example.

Depending on the package's internal connection resources all non-critical path cones pending to the critical cone path could be duplicated and merged into the critical path cone, for speed.

Non-critical path pending cones will be merged into the critical one based on a linear criterion computed using graph quality factors (amount of internal nodes in such a non-critical cone, number of internal connections, minimal delay introduced etc).



Non-critical path pending cones will be merged into the critical one based on a linear criterion computed using graph quality factors (amount of internal nodes in such a non-critical cone, number of internal connections, minimal delay introduced etc).

Fig. 1. Directed acyclic graph

However non-critical cones are considered in decreasing critical order and will be mapped separately and this will save area (CLBs) and internal interconnections resources. Mapping process was implemented using *minDepth* algorithm [6] and *minLevelMapII* algorithm derived from the previous one but with powerful additional heuristics as it was presented in [7].

5. Experimental results

Implemented cluster algorithm working with *minLevelMapIIv2* (technological mapping) was tested against *minDepth* used without cluster partitioning. Results are presented in Table 1.

Circuits, in Table 1, are taken from MCNC91 multilevel examples benchmark; being selected the most representative ones (as used in similar works). Cone partitioning algorithm is similar to those previously presented in literature, [5, 13, 15, 17, 18, 19], but modified to minimize first critical path delay. This was implemented by merging those clusters containing nodes belonging to the critical path but having enough *slack* in order to introduce no other costs to the partitioning objective.

Heuristics introduced to evaluate cone's costs are based on the published results, [13, 16, 19], but them are slight modified because actual application was exclusively targeted to map *k*-LUT based FPGAs having primary goal to find best performance circuit and, after that, area optimal solution. Cluster generation, is based upon algorithm illustrated in [6] and provided most of the application's backgrounds. Actual algorithm is computing all clusters *Clusters*(*n*) rooted on internal node *n* and having less inputs than *M* (M> *input*(*Clusters*(*n*)) in an efficient way compared to the method *MaxFlow-MinCut* used in most of the non-heuristic existing works, see [8, 9, 13, 24].

Comparing results for *minDepth* and *minLevelMapIIv2* it's obvious that almost all results are a little less adequate, in Table 1, for *minLevelMapIIv2* (upgraded *minDepth*) with cluster partitioning. However, these results are better than those previously reported in [27, 28], because several heuristics were improved.

That's because *minLevelMapIIv2* is still working, mainly on the non-critical path cones, under the cone's boundaries and is not always able to find best merging nodes with this restriction, while *minDepth* is working ignoring cones boundary restrictions and finds always best area results (even using less sophisticated heuristics for that).

Although a number of clustering algorithms, such as the random walk based clustering algorithms, [18, 9], the clique based method [14], and the multi-commodity-flow based method [26], have been developed most of them are not considering signal flow during cluster generation and finally cluster mapping.

Table 1.

Comparative experimental results on MCNC91 multilevel benchmark

Circuit name	*minDepth*		*MinLevelMapIIv2 with clustering*	
	depth	LUTcnt	depth	LUTcnt
5xp1	2	21	2	22
9sym	5	7	5	8
C499	4	67	4	68
C5315	8	500	8	503
C880	7	130	7	136
alu2	5	129	5	131
alu4	5	549	5	550
apex2	5	150	5	153
apex4	5	875	5	885
apex6	4	222	4	225
apex7	4	67	4	72
b9	3	37	3	40
bw	1	28	1	30

clip	3	44	3	45
count	3	74	3	74
des	5	1014	5	1022
duke2	4	151	4	153
e64	3	338	3	343
f51m	3	51	3	51
misex1	2	17	2	17
misex2	2	42	2	43
rd73	2	8	2	8
rd84	3	13	3	13
rot	6	204	6	209
sao2	4	57	4	57
vg2	3	35	3	35
z4ml	2	5	2	5

6. Conclusions and future work

Existing cluster-based partitioning approaches have reported consistent improvements, in terms of both the cut size and the run time, over direct partitioning on the initial circuit. Since fully automatic partitioning is essential for fast iterations in the design cycle, considerable effort is made in academia as well as in industry to facilitate and improve the difficult decisions on functional level.

Both mapping algorithms are, actually, under research and development in order to be able to accept various and complex *delay models* together with new mapping heuristics in order to obtain better *area* results.

Cluster partitioning algorithm, also under development, will be enhanced with new fast cost estimators making more efficient non-critical path cones process. Additional to the technological mapping of FPGA circuits, cluster-partitioning algorithm, has applications in large decision diagrams partitioning.

R E F E R E N C E S

1. *B. Kernighan, S. and Lin*, An Efficient Heuristic Procedure for Partitioning of Electrical Circuits, in Bell System Technical Journal, February 1970.
2. *R. Boppana,* Eigenvalues and Graph Bisection: An Average-Case Analysis, in *IEEE Symosium on Foundations of Computer Science*, pp. 280-285, 1987.
3. *K. R. Azegami, M. Inagi, A. Takahashi, and Y. Kajitami,* An Improvement of Network-Flow Based Multi-Way Circuit Partitioning Algorithm, in IEICE Transactions on Fundamentals, **Vol. E85-A**, No.3, pp. 655-663, March 2002.
4. *M. Beardslee, and A. Sangiovanni-Vincentelli*, Heuristic Methods for Communication-Based Logic Partitioning, in *4th ACM/SIGDA Physical Design Workshop*, pp. 199-210, April 1993.
5. *D. Brasen, , and G. Saucier*, FPGA Package Partitioning for Performance, in Proceedings of the '94 FPGA Symposium, Section 1, Field-Programmable Systems, Posters, 1994.
6. *I. Bucur*, An Optimal Technology Mapping for Delay Optimization of Lookup Table-Based FPGAs, in Proceedings of the 12th International Conference on Control Systems and Computer Science, Bucharest, Romania, May 26-29, 1999.
7. *I. Bucur*, An Optimal *k*-Clustering in Directed Acyclic Graphs, in Proceedings of the Sixth International Conference on Economic Informatics, pp. 364 -368, May 2003.
8. *P. Chan, M. Schlag, and J. Zien*, Spectral K-Way Ratio-Cut Partitioning and Clustering, in Proceedings of the 30th ACM/IEEE Design Automation Conference (DAC'93), 1993.
9. *J. Cong, and Y. Ding,* On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping, in Proceedings of the 30th ACM/IEEE Design Automation Conference (ICCAD'93), pp. 213-218, 1993.
10. *J. Cong, L. Hagen, and A. Kahng,* Random Walks for Circuit Clustering, in *Proceedings of the IEEE 4th International ASIC Conference*, Sept. 1991, pp. P14-2.1, 1991.
11. *J. Cong, L. Hagen, L., and A. Kahng,* Net Partitions Yield Better Module Partitions, in *Proceedings of the IEEE 29th Design Automation Conference* (*DAC'92*), pp. 47-52, 1992.

12. *J. Cong, Z. Li, and R. Bagrodia,* Acyclic Multi-way Partitioning of Boolean Networks, in Proceedings of the *31st* Design Automation Conference (DAC'94), pp. 670 – 675, 1994.
13. *J. Cong, J., and S.K. Lim*, Edge separability based circuit clustering with application to circuit partitioning, in Proceedings of the IEEE/ACM Asia South Pacific Design Automation Conference (SPDAC-2000), pp. 429—434, 2000.
14. *J. Cong, J., M.A. Smith,* Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Designs, in Proceedings of the ACM/IEEE Design Automation Conference (DAC'93), pp. 755-760, 1993.
15. *A. Dasdan, and C. Aykanat*, Improved Multiple-way Circuit Partitioning Algorithms, in Proceedings of the 1994 FPGA Symposium, Section 1: Field-Programmable Systems, Posters, 1994.
16. *C. Fiduccia, and R. Mattheyses,* A Linear Time Heuristic for Improving Network Partitions, in Proceedings of the ACM / IEEE Design Automation Conference (DAC'82), pp. 175-181, 1982.
17. *L. Hagen, L., A.B. Kahng,* Fast spectral methods for ratio cut partitioning and clustering, in Proceedings of International Conference on Computer-Aided Design (ICCAD'91), pp. 10 – 13, 1991.
18. *L. Hagen, A. and A.B. Kahng*, A New Approach to Effective Circuit Clustering, in Proceedings of the International Conference on Computer-Aided Design (ICCAD'92), pp. 422-427, 1992.
19. *L. Hagen, and A.B. Kahng,* New Spectral Methods for Ratio Cut Partitioning and Clustering, in *IEEE Trans. on CAD*. **Vol. 11,** No. 7, pp. 1074-1085, July 1992.
20. *J. Hwang, and A.E. Gamal,* Optimal Replication for Min - Cut Partitioning, in Proceedings of the International Conference on Computer-Aided Design (ICCAD'92), pp. 432-435, November 1992.
21. *S. Iman, M. Pedram, C. Fabian, and J. Cong,* Finding Uni-Directional Cuts Based on Physical Partitioning and Logic Restructuring, in Proceedings of the 4th ACM/SIGDA Physical Design Workshop, pp. 187-198, 1993.
22. *F.M. Johannes*, Partitioning of VLSI Circuits and Systems, in *Proceedings of the 33rd Annual Conference on Design Automation* (*DAC'96*), pp. 83-87, 1996.
23 *C. Kring, and A.R. Newton*, A Cell-Replicating Approach to Mincut-Based Circuit Partitioning, in *Proceedings of the International Conference on Computer-Aided Design* (*ICCAD'91*), pp. 2-5, 1991.
24. *H. Krupnova, A. Abbara, and G. Saucier*, A Hierarchy-Driven FPGA Partitioning Method, in 34th Conference on Design Automation Conference, (DAC'97) Pages: 522-525, 1997.