

# Large-scale Nano-PLA Meshes with Efficient Logic Mapping

•  
•

---

## Architecture and Max-Clique Partitioning Algorithm

Kumar Jeev

Department of Computer Science  
Coastal Carolina University  
Conway, SC 29528-6054 , USA  
kjeev@coastal.edu

**Abstract.** An area efficient architecture for organization of sublithographic PLAs(Programmable Logic Arrays) is proposed. Two-way max-clique clustering and partitioning algorithm is suggested for use on this architecture for efficient logic mapping. This algorithm bridges the barrier between the traditional schemes of Top-down partitioning and Bottom-up clustering in VLSI design and gives a unified approach to the problem of logic mapping. The scheme is based on recent developments in heuristics for maximum clique search in an undirected graph using sequential coloring.

## 1 Introduction

Recent advances in molecular electronics [27] [17] and sub-lithographic Nano-architectures of FPGAs(Field Programmable Gate Arrays) [10] [16] [33] has reaffirmed the belief that nanotechnology will extend the Moore's Law [28] in this new century [19] [30]. Most of these nano-architectures [3] [11] [12] [32] are crossbar structures because of their ease of fabrication and support for defect tolerance. This paper envisions an architecture for full-scale electronic systems based on the crossbar approach.

We organize the nano-PLAs into nano-PLA blocks and further group them into nano-PLA clusters. These nano-PLA clusters form the building blocks of reconfigurable nano-electronic devices similar to the existing lithographic CPLDs (Complex Programmable Logic Devices). Because of the bottom-up fabrication techniques used to make these nano-electronic devices (nano-CPLDs), they will

have interconnect requirements very different [12] from the existing CPLDs [13]. Their efficiency would closely depend on the logic mapping technique used to embed logic onto them.

Traditionally, logic mapping techniques in VLSI design literature have been divided into two classes - the Bottom-up Clustering and the Recursive Top-down Partitioning [26]. The Top-down partitioning methods [24] [14] [34] divide a circuit into smaller and more manageable components. The Bottom-up clustering methods [6] [18] [31] look at the problem of circuit logic break-up from the view of grouping together elements to form circuit clusters. While top-down methods fail in cases when the circuits are large because of their complexity, bottom-up fail because of their lack of reliance on the global view of the circuit. The algorithm proposed banks on the strengths of both these methods and in that sense is a hybrid of the two.

Recent developments in graph heuristics for maximum clique search [8] [22] are the motivation of the proposed algorithm. These heuristics are used to find the maximum clique in a circuit. This clique is removed and the maximum clique in the residual graph is searched and so on. The partitions so obtained are unbalanced and we balance them using bottom-up techniques based on density criterions.

We thus makes two original contributions through this paper. First, we suggests a feasible architecture for full-scale nano-devices based on Programmable Logic Array Blocks. Second, we propose an integrated algorithm based on top-down and bottom-up techniques to map logic into these systems.

The paper is organized into four sections. The first section gives an overview of the proposed architecture. The next section discusses the concepts and definitions used in the algorithm for logic mapping onto this architecture. The third section presents the algorithm. In the fourth section we analyze the results of our algorithm on a benchmark circuit. Finally, we state the conclusions and avenues of further research.

## 2 Architecture

Large PLA blocks provide higher speeds and guarantees predictable timing of operations. At the same time they increase the complexity of logic synthesis because of the NP-hard complexity of the two level minimization problem [8]. However, it has been shown [5] that the PLAs outperform LUT-based FPGAs in terms of both delay and area. This is why most of the new architectures using sublithographic techniques aim at making nano-PLAs [3] [12] [32]. The nano-CPLDs that will result from these PLAs will have very high logic gate density and will require very effective CAD tools to achieve better performance and to maximize logic utilization similar to that required by some of the new CPLDs [7].

Advances in controlled nanowire construction [9] [29], assembling them into crossbar structures [20], establishing diode crosspoints [3], and techniques for addressing nanowires from lithographic scale wires [11] have established crossbar

structure a feasible architecture for nano-PLAs. There have been several variants of these nano-PLAs [12] [16] [33].

To make full scale nano-CPLDs from these nano-PLAs we propose the following hierarchy. The nano-PLAs are first organized into nano-PLA blocks which are PLAs wired along with a nano-Buffer/Inverter Unit [12]. These nano-PLA blocks are then organized in such a way that upon logic mapping they differentiate into nano-PLA clusters. We will demonstrate in the last section that logic mapping attains its best packing efficiency at different cluster sizes depending on the logic. This cluster size is defined as the natural cluster size for a circuit. These clusters together form the entire nano-CPLD. Making cluster size dependent on the logic mapped onto the nano-CPLD ties the architecture closely with the logic mapping algorithm and provides maximum logic packing efficiency. In the next section, we introduce the techniques used by our algorithm for logic mapping on these nano-CPLDs.

### 3 Mathematical Framework

Circuit Partitioning and Clustering are the two approaches of dividing a circuit into group of nodes which are densely interconnected as compared to the rest of the graph thus reducing the interconnect. In Computer Aided Design literature for VLSI logic mapping these two are referred to as the Top-down (Partitioning) and Bottom-up (Clustering) approaches.

Top-down approaches partitions a given circuit into smaller partitions. These techniques have the advantage of having a complete view of the circuit but at the same time the disadvantage in terms of computational resource requirement due to the need for global analysis at each step [31]. The top-down approach includes the iterative bi-partitioning algorithms like the Fiduccia-Mattheyses [14] variant of the Kernighan-Lin algorithm [24], flow oriented repeated minimum-cut algorithms [36], etc.

Bottom-up approaches cluster nodes in a given circuit based on some clustering criterion into small clusters. These techniques have the advantage of being fast but at the same time they have more probability of making mistakes and hence waste time in back-tracking [31]. The bottom-up approaches include greedy clustering methods, density based methods [21], random walk method [4], techniques using transitive closure (k-1 connectedness [15]), etc. All these algorithms use several different clustering objectives to merge nodes into cluster, each one has its own pros and cons which have been studied in great detail [23].

In our approach we take the problem of circuit partitioning into a graph theoretic framework. Recently, it has been shown that all real life graphs are 1-perfect [8]. Thus, it is easy to color them and find the maximal clique. There are several heuristics to find the maximum clique [8] [22]. Some people have suggested the use of coloring algorithms to cluster circuits [31].

We iteratively divide the circuit on the basis of maximal cliques (which are arguably the densest regions in a circuit) using heuristics (top-down partitioning). Then we recombine the smaller cliques into clusters using some clustering

criterion (bottom-up techniques). This technique thus takes the benefit of the advantages of both the bottom-up techniques and the top-down techniques.

These clusters are then mapped onto the nano-PLAs using the traditional mappers like DDMMap [6], TEMPLA [1], PLAMap [2], etc. Most of these mappers assume that the input network has been decomposed to 2-bounded network and so our clusters fit easily into these algorithms.

We model the boolean circuit as an undirected graph  $G(V, E)$ , where the vertex set  $V = \{v_i | i = 1, 2, 3, \dots, n\}$  represents the gates(nodes) while the edge set  $E = \{e_{i,j} | i, j = 1, 2, 3, \dots, n\}$  represents the fact that  $i$  and  $j$  are input-output pairs. A hyperedge from a netlist(list of nets or hyperedges)  $N$  with source  $n$ :  $N_n = \{n, n_1, n_2, n_3, \dots, n_n\}$  is modeled as the set of edges  $\{e_{n,n1}, e_{n,n2}, e_{n,n3}, \dots, e_{n,nn}\}$  thus  $n$  and  $n_k$  for  $k = 1, 2, 3, \dots, n$  form input-output pairs. We store the graph in the form of an adjacency matrix  $A = [a_{ij}]$  where  $i$  and  $j$  are input-output pairs.

A *cluster* is a sub-graph of the boolean circuit graph. A *maximal clique* is the maximum sized complete connected sub-graph. We will now state the proposed algorithm in the next section.

## 4 Max-Clique Partitioning Algorithm

Our algorithm takes its input from a netlist. We convert this netlist into an undirected graph  $G$  using the model developed in the previous section. We use Coudert's heuristics [8] to find the maximal clique  $C$  in this graph. We then remove the maximal clique from the graph and again use the heuristics to find the maximal clique in the residual graph  $G-C$ . This process continues until we have partitioned the entire graph into partitions which are cliques of different sizes.

Although graphs with high edge density have a balanced breakup in terms of the number of partitions of different sizes (Table 1). The picture changes completely when we look at sparse graphs similar to the ones observed in real-life circuits (Table 2). Thus in practice the size of the partitions reduces very fast as the algorithm progresses and a major chunk of these partitions are single nodes or edges (two nodes).

This is where we introduce the bottom-up techniques to regroup these cliques into larger clusters. We set a window for the allowed size of a partition based on the size of the maximal clique for the graph. Then we regroup these clusters. There can be several different criteria for doing this regrouping. In the criteria that we use the smallest cluster is regrouped with the cluster with which it shares the maximum number of edges. This criteria helps bring down the interconnect in the partition. This process continues until all the clusters (except maybe one) are of sizes within that window. When we do this for the case of Table 2 then we get Table 3.

The only problem now is that these clusters are very small and far too many. To resolve this issue we form a adjacency matrix from these clusters, with an edge between two clusters representing the existence of an edge between one of

**Table 1.** The number of partitions of different sizes generated by repeated max-clique partitioning on a 256 random graph generated with 10% probability of having edges between two vertices.

Size of Partition	Number of partitions
1	11
2	11
3	17
4	29
5	10
6	1

**Table 2.** The number of partitions of different sizes generated by repeated max-clique partitioning on a 256 random graph generated with 1% probability of having edges between two vertices.

Size of Partition	Number of partitions
1	66
2	77
3	8
4	3

**Table 3.** The number of partitions of different sizes generated by repeated max-clique partitioning with re-clustering (with acceptable partition size 3-5) on a 256 random graph generated with 1% probability of having edges between two vertices.

Size of Partition	Number of partitions
1	0
2	1
3	74
4	8

the nodes in the cluster and the other in the second cluster. Then we perform max-clique partitioning on this new graph. The new partitions generated are partitions of clusters and are then converted into partitions of nodes. We again re-cluster them into cluster with a new window of accepted cluster sizes. This goes on until we find clusters of reasonable sizes which impose reasonable interconnect requirements.

The cluster sizes that we obtain in this fashion are in some sense the natural sizes into which the graph should be partitioned because they are based on both the global conditions of edge density as well as the local ones of interconnect sparsity. This is why in our proposed architecture emphasizes the need for re-configurable cluster sizes which is dependent on the partitioning performed by our algorithm.

In the next section we show the results obtained by running our algorithm on a benchmark circuit.

## 5 Results

We run our algorithm on the "apex4" which is a fairly large MCNC (Microelectronics Center of North Carolina) Benchmark circuit [35] with 1147 nodes and 527 nets. This circuit has an unpartitioned circuit area (total number of edges in adjacency graph) of 2549. We perform four iterations of our algorithm on this circuit. The total number of clusters decreased with each iteration while the maximum number of nodes in a cluster increased as in Table 4.

**Table 4.** The number of clusters and the maximum number of nodes in a cluster in every iteration of run on apex4.

Iteration	Number of Clusters	Maximum number of nodes in a Cluster
1	337	4
2	100	14
3	15	105
4	2	938

We define the *cluster area* as the number of edges between nodes within a cluster. The *total cluster area* is the sum of the area of all the clusters and the *average cluster area* is the average of the area of all the clusters. We observe that all these quantities increase with each iteration as in Table 5.

Finally, we look at the interconnect that we get from the partitioning and clustering. We define *interconnect* as the total number of edges in the cluster adjacency graph. Thus if there are more than one edges in the graph of the circuit between nodes in one cluster and those in another then they contribute just one unit to the interconnect. Similarly if there is no edge between nodes in one cluster and those in another then there is no contribution to the interconnect by

**Table 5.** The total cluster area, maximum cluster area and the average cluster area in every iteration of run on apex4.

Iteration	Total Cluster Area	Maximum Cluster Area	Average Cluster Area
1	335	6	0.994065
2	545	21	5.45
3	810	134	54
4	2158	2012	1079

that cluster pair. *Cluster to Cluster Interconnect(C2CI)* for a cluster is the total number of interconnect edges it shares with different clusters. The *Partitioned Circuit Area(PCA)* is a function of the interconnect(I), the maximum circuit area(MCA) and the number of clusters(N). Mathematically,

$$PCA = I + MCA * N \quad (1)$$

We observe in our results that although the Interconnect and the Maximum Circuit to Circuit Interconnect reduces with each iteration, the Partitioned Circuit Area first decreases and then increases (Table 6). The maximum circuit size at the iteration when the PCA is minimum is arguably the ideal cluster size for a nano-PLA cluster.

**Table 6.** The interconnect, maximum cluster to cluster interconnect(C2CI) and the partitioned circuit area in every iteration of run on apex4.

Iteration	Interconnect	Maximum C2CI	Partitioned Circuit Area
1	1795	71	3817
2	1002	43	3102
3	99	7	2109
4	1	1	4024

## 6 Conclusions and Future Work

We have proposed that efficient architectures of nano-CPLDs should have nano-PLAs grouped into a nano-PLA clusters which should have a reconfigurable maximum cluster size. We have also shown one possible alternative for an algorithm for both finding the optimal cluster size and also the clusters themselves. The algorithm successfully exploits the strengths of both the top-down and bottom-up approaches. Further research needs to be done in comparing this algorithm with some of the existing algorithms.

## Acknowledgements

I am thankful to Dr. Andre DeHon and Ms. Helia Naeimi for their support and guidance during the project. I am also thankful to Caltech Department of Computer Science for inviting and sponsoring me for the Computing Beyond Silicon Summer School 2004.

## References

1. Anderson J. H., Brown S. D., "Technology mapping for large complex PLDs", Proceedings of the 35th Annual Design Automation Conference, San Francisco, California, United States, p.698-703, June 15-19, 1998.
2. Chen D., Cong J., Ercegovic M. D., and Huang Z., "Performance-Driven Mapping for CPLD Architecture", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, no. 10, pp. 1424-1431, October 2003.
3. Chen Y., Jung G., Ohlberg D., Li X., Stewart D., Jeppesen J., Nielsen K., Stoddart J., Williams R., "Nanoscale molecular-switch crossbar circuits", Nanotechnology, 14:4628, 2003.
4. Cong J., Hagen L. W., Kahng A.B., "Random Walks for Circuit Clustering", Proceedings of the IEEE Conference on ASIC, pp. 14.2.1 - 14.2.4, June 1991
5. Cong J., Huang H., Yuan X., Technology mapping for k/m-macrocell based FPGAs, in Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays, San Jose, CA, Feb 2000, pp. 5159.
6. Cong J., Smith M., "A Parallel Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design", Proceedings of the Thirtieth IEEE Design Automation Conference, pp. 755-560, 1993.
7. Cong J., Xu S., "Synthesis challenges for next-generation high-performance and high-density PLDs", Proceedings of the 2000 conference on Asia South Pacific design automation, p.157-162, January 2000, Yokohama, Japan.
8. Coudert O., "Exact Coloring of Real-Life Graphs is Easy", Proceedings of Thirty-fourth IEEE Design Automation Conference, (00):121-126, 1997.
9. Cui Y., Lauhon L. J., Gudiksen M. S., Wang J., Lieber C. M., "Diameter-controlled synthesis of single crystal silicon nanowires" Applied Physics Letters, 78(15):22142216, 2001.
10. DeHon A., "Array-Based Architecture for FET-based Nanoscale Electronics", IEEE Transactions on Nanotechnology, 2(1):23-32, March 2003.
11. DeHon A., Lincoln P., Savage J., "Stochastic Assembly of Sublithographic Nanoscale Interfaces", IEEE Transactions on Nanotechnology, 2(3):165-174, 2003.
12. DeHon A., Wilson M. J., "Nanowire-Based Sublithographic Programmable Logic Arrays", Proceedings of the 2004 ACM/SIGDA Twelfth International Symposium on Field Programmable Gate Arrays, pp. 123-132, February 2004.
13. DeHon A., "Balancing interconnect and computation in a reconfigurable computing array", Proceedings of the 1999 ACM/SIGDA Seventh International Symposium on Field Programmable Gate Arrays, pp. 69-78, February 1999.
14. Fiduccia C. M., Mattheyses R. M., "A Linear-Time Heuristic for Improving Network Partitions", Proceedings of Nineteenth Design Automation Conference, pp. 175-181, 1982.
15. Garbers J., Promel H.J., Steger A., "Finding Clusters in VLSI Circuits", ICCAD 1990, pp. 520-523.

16. Goldstein S. C., Budiu M., "NanoFabrics: Spatial Computing using Molecular Electronics", Proceeding for International Symposium on Computer Architecture, pp. 178-189, June 2001.
17. Gudiksen M. S., Lathon L. J., Wang J., Smith D. C., Lieber C. M., "Growth of nanowire superlattice structures for nanoscale photonics and electronics", *Nature*, 415:617620, February 7 2002.
18. Hagen L., Kahng A. B., "A New Approach to Effective Circuit Clustering", Proceedings of IEEE/ACM International Conference on Computer-Aided Design, pp. 422-427, 1992.
19. Heath J. R., Kuekes P. J., Snider G. S., Williams R. S., "A defect-tolerant computer architecture: Opportunities for nanotechnology", *Science*, 280:1716-1721, June 12, 1998.
20. Huang Y., Duan X., Wei Q., Lieber C. M., "Directed assembly of one-dimensional nanostructures into functional networks", *Science*, 291:630633, January 26 2001.
21. Huang D. J.-H., Kahng A. B., "When clusters meet partitions: new density-based methods for circuit decomposition", Proceedings of the 1995 European conference on Design and Test, p.60, March 06-09, 1995
22. Jagota A., Sanchis L., "Adaptive, Restart, Randomized Greedy Heuristics for Maximum Clique", *Journal of Heuristics*, 7:6, pp. 565-585, November 2001.
23. Kahng A. B., Sharma R., "Studies of Clustering Objectives and Heuristics for Improved Standard-Cell Placement", UCLA CS Dept report, January 1997.
24. Kernighan B. W., Lin S., "An efficient heuristic procedure for partitioning graphs", *Bell System Technical Journal*, 49:291-307, February 1970.
25. Kouloheris J. L., Gamal A. E., "FPGA Performance vs. Cell granularity", Proceedings of the Custom Integrated Circuits Conference, 1991, pp.621-624.
26. Lengauer T., "Combinatorial algorithms for Integrated Circuit Layout", John Wiley & Sons, Chichester, 1990.
27. Luo Y., Collier P., et.al., "Two-Dimensional Molecular Electronics Circuits", *ChemPhysChem*, 3:519-525, 2002.
28. Moore G., Cramming more components onto integrated circuits, *Electronics*, Vol. 38, No. 8, April 19, 1965.
29. Morales A. M., Lieber C. M., "A laser ablation method for synthesis of crystalline semiconductor nanowires", *Science*, 279:208211, 1998.
30. Milunovich S., Roy J. M. A., "The Next Small Thing - An introduction to nanotechnology", Merrill Lynch Technical Trends Comment, United States Technology Strategy, September 2001.
31. Singh A., Marek-Sadowska M., "Circuit clustering using graph coloring", Proceedings of International Symposium Physical Design, pp. 164-169, 1999.
32. Snider G., Kuekes P., Williams R. S., "CMOS-like logic in defective, nanoscale crossbars", *Nanotechnology*, 15:881891, 2004.
33. Tour J. M., "Molecular Electronics: Commercial Insights, Chemistry, Devices, Architecture and Programming", World Scientific Publishing Company, New Jersey, 2003.
34. Wei Y. C., Cheng C. K., "Towards Efficient Hierarchical Designs by Ratio Cut Partitioning, Proceedings of IEEE International Conference on Computer-Aided Design, pp. 298-301, 1989.
35. Yang S., Logic Synthesis and Optimization Benchmarks, Version 3.0, Tech. Report, Microelectronics Centre of North Carolina, 1991.
36. Yeh C. W., Cheng C. K., Lin T. T. Y., "Circuit Clustering Using a Stochastic Flow injection Method", *IEEE Transactions on Computer-Aided Design*, Vol 14 No 2, pp. 154-62, February 1995.