The Effect of LUT and Cluster size on a Tree based FPGA Architecture

Umer Farooq, Zied Marrakchi, Hayder Mrabet and Habib Mehrez
LIP6, Université Pierre et Marie Curie
4, Place Jussieu, 75005 Paris, France
umer.farooq@lip6.fr

Abstract

In this paper we present the effect of lookup table (LUT) size (no of inputs to a LUT) and cluster size (no of LUTs per cluster) on the area and critical path of a tree based FPGA architecture (MFPGA) [12]. For this purpose we have designed a flow that places and routes a set of bench mark circuits on different tree based architectures with varying lookup table (lookup table size varies from 3 to 7) and cluster sizes (cluster size varies from 4 to 8). With the help of experimental results we have analyzed the alteration in the MFPGA area with different LUT and cluster sizes. We have shown that in general LUTs having 4 or 5 inputs and clusters having 4 or 5 LUTs per cluster produce most efficient results in terms of area for the tree based architecture. We have also determined the mutation in the number of switches crossed by the critical path with changing LUT and cluster size and we have shown that architectures with higher LUT size and higher cluster size can be more optimal in terms of critical path though they are not optimal in terms of area.

1 Introduction

Many studies in the past several years have been carried out to see the effect of lookup table(LUT) and cluster size on the density of FPGA architecture. The work in [3] compiles a very detailed study regarding the effect of LUT and cluster size on the density and performance of FPGA architecture. In [3] the authors have shown that LUTs with sizes 4 to 6 and clusters with sizes 3 to 10 give the most efficient results in terms of area-delay product for an FPGA. The work in [14] demonstrated that LUT size of 4 is most area efficient in a non clustered context. The work in [8] suggests that architecture with heterogenous mixture of LUT sizes 2 and 3 is as area efficient as the architecture with LUT size 4. The question of effect of LUT and cluster size on FPGA architecture is addressed in [6] and [9] also. But all the work previously done in this context focusses on the MESH architecture and no prior work has been done yet for tree based architectures.

In [7] a novel tree based architecture is presented where authors show that this architecture is better in terms of density as compared to common VPR-style MESH architecture [4]. This architecture is further improved in [12] in terms of FPGA density and routability. But this work considers LUTs with 4 inputs and one output only. A study of effect of variation of LUT size on the the density has not been performed yet for tree based architecture. The objective of this paper is to determine the effect of number of inputs to the LUT (K) and the number of LUTs in a cluster (N) on the area of tree based architecture.

Usually when lookup table or cluster size is increased, this gives rise to the functionality of cluster. This increase in the functionality reduces the overall number of clusters that are required to implement a certain function and also decreases the number of such blocks on the critical path. But on the other hand the area of clusters increases with increase in the LUT (K) and cluster size (N). The area of LUT increases with K due to increase in number of inputs to the LUT and area of cluster increases with N due to increase in number of inputs and outputs of the cluster. So an increase in LUT or cluster size may decrease critical path length and it may increase the cluster functionality but it can affect the area in a bad manner. So while exploring these tradeoffs we try to analyze the effect of LUT and cluster size on the area of tree based architecture.

This paper is outlined as follows: Section 2 describes the interconnect architecture of tree based FPGA. Section 3 describes the experimental configuration flow that we use in our experiments and the steps that are performed to estimate the area of tree based architecture. Section 4 outlines the different experimental results and section 5 concludes this paper and discusses some future work.

2 Interconnect Architecture

In [12] the authors have presented a tree based FPGA architecture. This architecture unifies two unidirectional

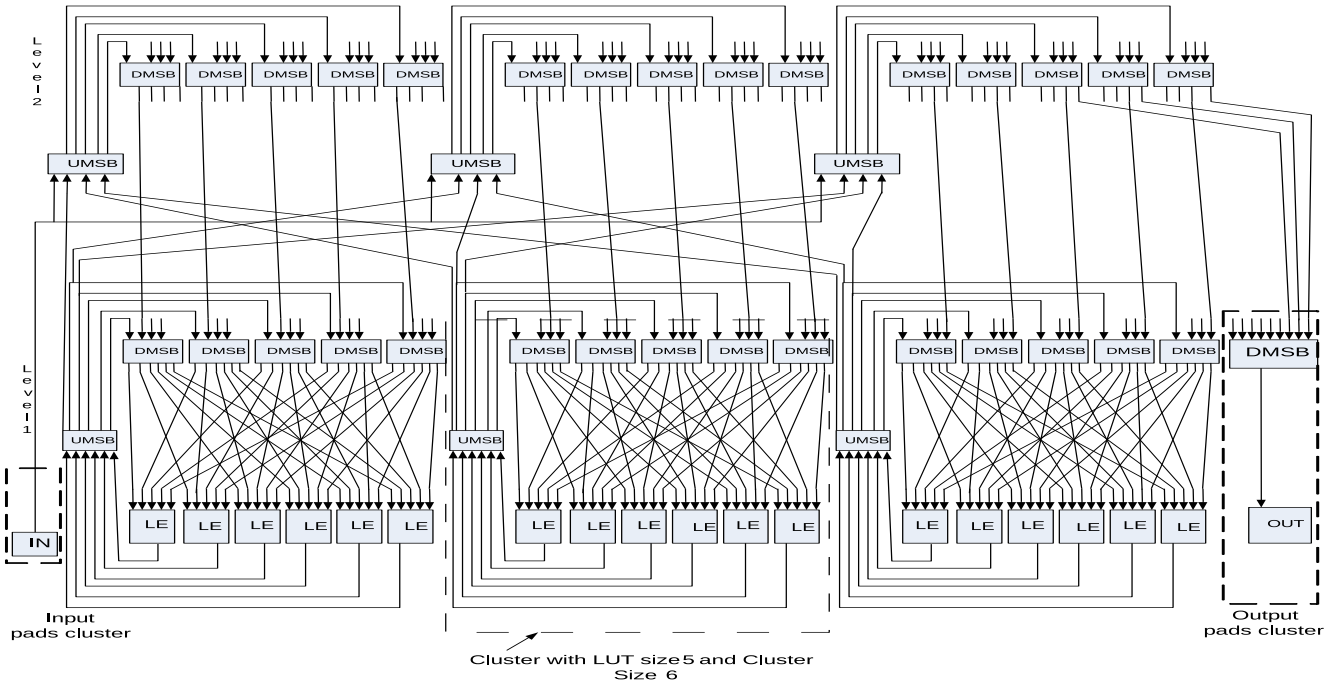

Figure 1. Tree-based interconnect: upward and downward networks

networks: A predictible downward network based on the Butterfly-Fat-Tree topology, and an upward network using hierarchy. The downward network comprises of downward mini switch boxes (DMSBs) that allow the connection between switch boxes and logic element (LE) inputs. The upward network comprises of upward mini switch boxes (UMSBs). These UMSBs allow LEs outputs to reach a larger number of downward mini switch boxes (DMSBs). The UMSBs are organized in a way that allows logic elements (LEs) belonging to the same "owner cluster" (at level 1 or above) to reach exactly the same set of DMSBs at each level. This upward network offers more routing paths to connect a source net to a given sink. So in this architecture a highly congested netlist is more likely to route because of the fact that logic elements (LE) in this architecture can reach a higher number of DMSBs per level. But the addition of UMSBs in the upward network causes an increase in the number of switches. However this increase in number of switches can be compensated by controlling the bandwidth of in/out signals of clusters at each level. The bandwidth of clusters is controlled using Rent's rule [10] which is easily adapted to tree based architecture. This rule states that

$$IO = k.n^{\ell.p} \tag{1}$$

where ℓ is a tree level, n is the cluster size, k is the number of in/out pins of a LUT and IO is the number of in/out pins of a cluster at level ℓ. p is the factor which controls the locality in interconnect at each level of the tree based architecture. In a tree based architecture both the upward and downward interconnect resources are controlled with the help of p. If the most of the connections are restricted locally, then the value of p can be reduced. But a reduction in its value affects the routabiltiy of the circuit. So we can reduce the value of p only up to a certain limit. Also it is shown in [12] that the number of switches required per logic element (LE) for the tree based architecture are

$$N_{switch}(LE) = \begin{cases} O(1) & \text{if } p < 1 \\ O(\log_k(M)) & \text{if } p = 1 \end{cases} \tag{2}$$

where M is the total number of LEs. Number of switches required per logic element for common MESH architecture are

$$N_{switch}(LE) = O(M^{p-0.5}) \tag{3}$$

As we can see from equations 2 and 3 that the number of switches grow more slowly in the tree based architecture as compared to common MESH architecture thus it was concluded that this architecture was more efficient in terms of density as compared to MESH especially for large circuits. But this architecture [12] could support netlists with LUTs having four inputs and one output only. So in order to address the questions raised in section 1 we have made this architecture flexible and now it can support netlists with any number of LUT inputs and outputs. An example of the modified architecture with LUT size 5, cluster size 6 and p=0.61 is shown in the figure 1. In this example we have shown a depopulated tree architecture where the number of inputs

per cluster are reduced from 30 to 15, the number of outputs are reduced from 6 to 3 and the number of UMSBs are reduced from 6 to 3. If we consider it as a two level architecture then the number of switches are reduced from 888 (p=1) to 489 (p=0.61). Also it is shown in the figure that the input and output pads are grouped to specific clusters so that input pads can reach all the LUTs of the architecture and output pads can also be reached by all the LUTs from different paths. It is important to mention here that equation 2 is applicable to the modified architecture. So the argument of slow growth in the number of switches as compared to MESH architecture is valid for this modified architecture too.

3 CAD Flow

The best way to determine the effect of variation of LUT (K) and cluster size (N) on the density of a tree based architecture is to experimentally place and route a circuit on different tree based architectures with varying LUT and cluster sizes. For this purpose a CAD flow is designed as shown in the figure 2. As it can be seen from the figure that first of all circuit is optimized through technology independent logic optimization using ABC [1]. After that, technology mapping (which is a process that converts the logic expressions into a net list of K-input LUT) is performed using "if" and "fpga" mappers (which are based on the notion of priority cuts) of ABC. Then all the registers and K-input LUTs are packed using T-VPACK [11]. T-VPACK gives us the netlist which is then used for generating the placement file with the help of partitioner.

Partitioning plays a very important role in generating an optimized MFPGA architecture in terms of density and also in terms of delay. Because the way the lookup tables are partitioned between clusters of the architecture has an important effect on the way the routing resources are utilized. Thus a good partitioner distributes the lookup tables in such a way that inter-cluster communication is reduced (i.e. communication outside cluster). Because local connections are cheaper not only in terms of delay but also in terms of routability so this might result in a more depopulated interconnect architecture and hence overall reduction in area. We have used a top-down recursive partitioning approach which gives priority to global connectivity information. First higher level clusters are constructed and then each cluster is partitioned into sub-clusters until the bottom of hierarchy is reached. So by using this top-down approach we are able to depopulate the architecture more on higher levels which results in overall efficient routing of the architecture. To perform partitioning we used hMetis [5]. It generates a good solution in a short time because of its multiphase refinement approach. Once the partitioning process is over, each cluster is assigned a position inside its owner

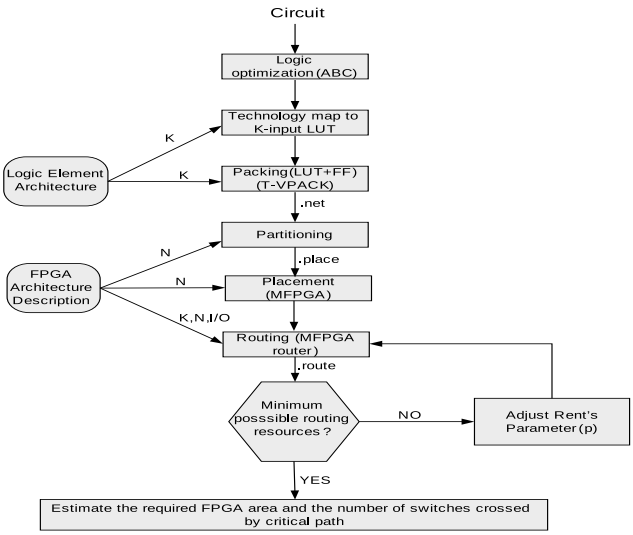

Figure 2. CAD Flow

cluster and these clusters are then placed on their respective positions during the placement phase of CAD flow.

After the placement, routing process is started. During routing nets are assigned to placed lookup tables so that they can communicate with the routing resources of the interconnect architecture. The routing algorithm that we have used is pathfinder [13], that uses an iterative negotiation based approach to successfully route all nets in a netlist. Two terminal nets are routed using Dijkstra's shortest path algorithm [15], and multi-terminal nets are decomposed into terminal pairs by the Prim's minimum-spanning tree algorithm [15]. At the end of an iteration, resources can be congested because multiple nets use them. During subsequent iterations, the cost of using a resource is increased, taking into account the number of nets that share the resource, and the history of congestion on that resource. Thus, nets are made to negotiate for routing resources.

Each time a circuit is routed, it is checked whether routing resources can still be minimized or not and then we try to route the circuit with more sparse resources. This process continues until an optimized architecture is obtained. The routing resources are minimized using Rent's parameter (p). As it is explained in section 2 that p controls the interconnect resources at each level, so we optimize the routing architecture in a level by level manner. This process is started at the bottom level and is continued until the top level of the architecture. Initially the value of p is set to 1 for all the levels and then circuit is routed. After that we decrement the value of p for bottom level only until the architecture becomes unroutable. Then we move towards the higher level and continue optimizing each level of the architecture until the top level of architecture is reached. So in this way we obtain an optimized routing architecture. Though this iter-

Cell	Area λ^2
sram	30×50
tri-state	35×50
buffer	20×50
flip-flop	90×50
mux 2:1	35×50

Table 1. Standard cells characteristics

ative optimization process is not applicable in real FPGAs but it gives us a measure of the demand of routing resources by each circuit for each architecture.

Once the routing process is over, area of the MFPGA required to implement the circuit is estimated using a refined estimation model of effective circuit area. We estimate the area as the sum of the area of lookup tables, area of flip-flops, area of all the associate programming bits and the area of multiplexers related to downward and upward interconnect. We use symbolic standard cells library [2] to estimate the FPGA required area. Different cells areas are presented in table 1.

4 Experimental Evaluation

In this section, the experimental results of the benchmark circuits are presented that are placed and routed on the MFPGA architecture using the CAD flow described in the section 3. In these experiments we have used twenty largest benchmark circuits from MCNC [16] along with ava from Altera. Table 2 gives a description of these benchmark circuits for a LUT size of 4. As it can be seen from the table that the number of lookup tables and the number of nets are more optimized as compared to the benchmarks in [3]. This is due to the fact that in this work we are using more efficient tools (ABC) for logic optimization and lookup table mapping as compared to the tools used in [3]. All of these circuits were placed and routed on the MFPGA architecture with LUT size ranging from 3 to 7 and cluster size ranging from 4 to 8. So with five different LUT sizes and 5 different cluster sizes we had a total of 25 distinct architectures for each of the benchmark circuit.

First of all we evaluate the experimental results that show how the area of a tree based FPGA varies with the change in the lookup table and cluster size (area is estimated using the model discussed in section 3). These results are shown in the figure 3. The results are based on the geometric average of the results obtained for benchmark circuits that are used in the experiments. It can be seen from the figure that initially there is a reduction in the average area between LUT size 3 and LUT size 4 and afterwards there is an increase in the area with a rise in the LUT and cluster size. It can be noted from the figure 3 that LUTs with 4 and 5 inputs are most area efficient for almost all cluster sizes and the architecture with cluster size 4 and LUT size 4 gives overall the most efficient average area for benchmark circuits.

When the cluster size is increased, more LEs are combined together in the same cluster and connections that would have been routed externally are now absorbed in the same cluster. This reduces the number of clusters required to implement the circuit and also the interconnect area which is an important part of the total area. But at the same time there is an increase in the area per cluster due to increase in

Circuit	No of 4-input LUTs	No of Nets
alu4	1242	1256
apex2	1522	1561
apex4	1089	1098
ava	10993	11006
bigkey	1147	1376
clma	5802	6218
des	1506	1762
diffeq	1161	1602
dsip	1145	1374
elliptic	2712	2845
ex1010	4143	4153
ex5p	982	990
frisc	2841	3747
misex3	1198	1212
pdc	3832	3848
s298	1091	1095
s38417	5190	5233
s38584	4696	4736
seq	1455	1496
spla	3045	3061
tseng	953	1005

Table 2. Description of Circuits used in Experiments (for K=4)

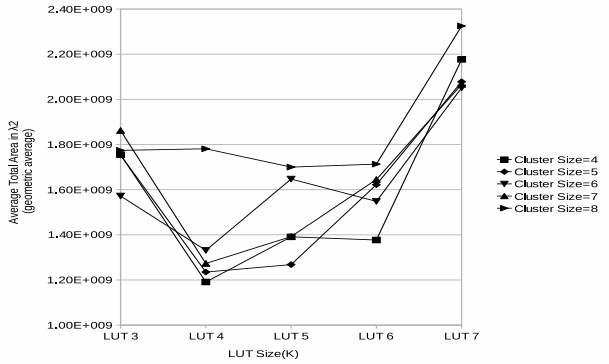

Figure 3. Total Area as a function of LUT and Cluster Size

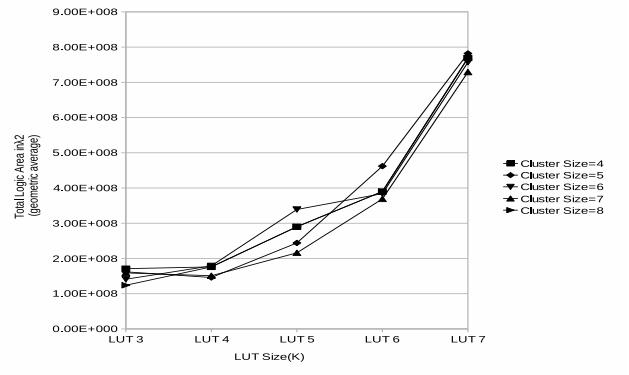

Figure 4. Total Logic Area as a function of LUT and Cluster Size

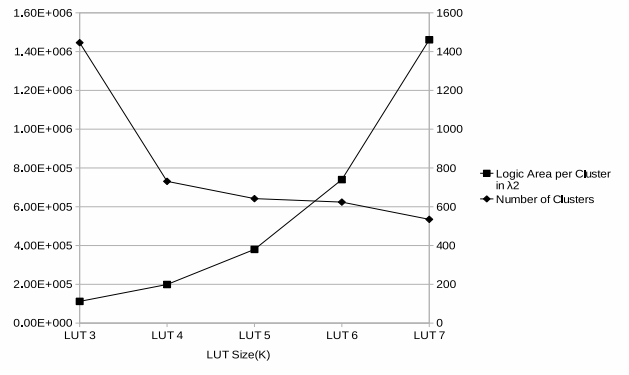

Figure 5. Number of Clusters and Cluster Area versus LUT Size (for N=4)

cluster size and increase in number of inputs and outputs of cluster. It should be noted that LE area also increases due to increased LUT size hence resulting in overall increased area.

In order to further explore the effect of LUT and cluster size on the area we divide the total area into two parts, one is called logic area and the other is called interconnect area. The total logic area comprises of the sum of the logic area of all the clusters at level 1 of the MFPGA architecture (Here we consider clusters at level 1 only because LEs are only at level 1 as can be seen from figure 1). Whereas the interconnect area comprises of the sum of area of all the interconnect resources of the architecture. The variation in the total logic area with LUT and cluster sizes is shown in the figure 4. It can be seen from the figure that there is an increase in the area with an increase in the LUT and cluster size. This trend can be explained with the help of figure 5. A plot of average logic area per cluster (primary Y-axis) and average number of clusters (secondary Y-axis) for a cluster size of 4 is shown in the figure 5. The behavior in this figure results due to the fact that with the increase in LUT size, the area of LUT increases exponentially hence there is an increase in the cluster area. Though there is a decrease in the number of clusters because each LE can implement more of the logic function. However the rise in the area is steeper as compared to decline in the total number of clusters so this results in the trend shown in figure 4.

Interconnect plays a key role in the total area of FPGA architecture. The effect of LUT and cluster size on the interconnect is shown in the figure 6. This figure shows that there is a decline in the interconnect area with an increase in lookup table size. The behavior in the figure 6 can be explained with the help of figure 7. In this figure we have plotted average interconnect area per cluster (primary Y-axis) and average number of clusters (secondary Y-axis) as a function of LUT size for a cluster size of 5. As it can be

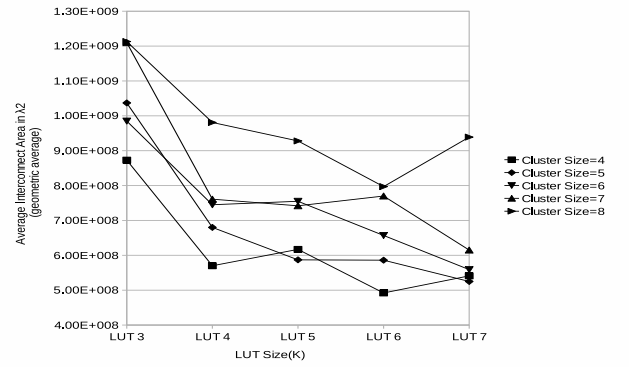

Figure 6. Interconnect Area as a function of LUT and Cluster Size

seen from the figure that with increase in LUT size there is an increase in interconnect area per cluster and there is a decrease in number of clusters. But the rise in interconnect area per cluster is not as steep as the decline in the number of clusters thus resulting in the behavior of figure 6. As we can see from figure 6 that decline in area is not as sharp as the rise in area of figure 4. So these two effects combined, result in the overall increase in the total area of the architecture as shown in figure 3.

The variation in the number of switches crossed by the critical path with the size of LUT and cluster is shown in the figure 8. Since we don't have information about layout, we only evaluate the number of switches crossed by critical path. It is clear from the figure that increase in the size of LUT or cluster decreases the number of switched crossed by critical path. This decrease in the number of switches is due to the fact that increase in LUT size or cluster size reduces the number of LUTs and the total number of clusters in the circuit. But at the same time there is an increase in the

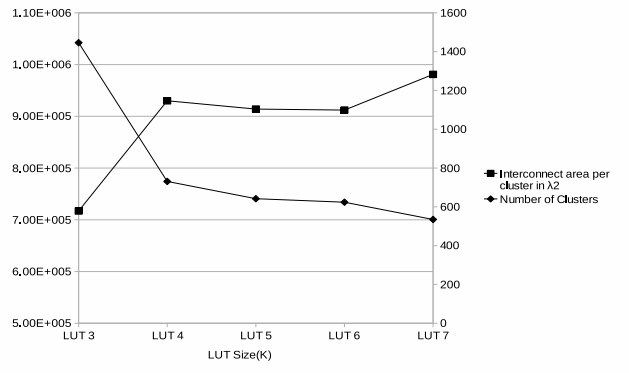

Figure 7. Number of clusters and Interconnect area per cluster versus LUT Size (for N=5)

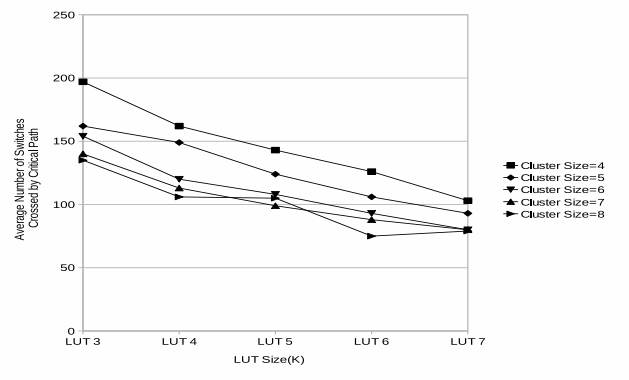

Figure 8. Number of Switches Crossed by Critical Path as a function of K and N

switch size because of the increased LUT and cluster size. Though the critical path delay is not presented in this work but from this figure we can state that with increase in LUT size or cluster size, there will be a decline in the critical path delay provided that increase in the internal delay of switch due to its increased size does not overshadow the decrease in number of switches.

5 Conclusion and Future Work

We have shown that in general LUTs with size 4 and 5 and cluster size 4 and 5 produce most efficient results in terms of area for the tree based FPGA. We have also determined the mutation in the number of switches crossed by the critical path as a function of LUT and cluster size and we have shown that LUTs with higher input size and with higher cluster size can be more optimal in terms of performance though they are not very good in terms of density.

In this work we have not performed timing analysis of the MFPGA architecture. So in future we want to perform the timing analysis of tree based FPGA so that we can determine the effect of LUT and cluster size on the performance of MFPGA and hence determine the best values of LUT and cluster size in terms of area-delay product.

References

[1] Berkeley logic synthesis and verification group,ABC: A system for sequential synthesis and verification, release 70930.http://www.eecs.berkeley.edu/ alanmi/abc/.

[2] A.Greiner and F.Pecheux. Alliance: A complete set of CAD tools for teaching VLSI design. *Eurochip Workshop*, 1992.

[3] E. Ahmed and J. Rose. The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density. *IEEE*, 2003.

[4] V. Betz, A. Marquardt, and J. Rose. Architecture and CAD for Deep-Submicron FPGAs. *Kluwer Academic Publishers*, January 1999.

[5] G.Karypis and V.Kumar. Multilevel k-way hypergraph partitioning. *Design automation conference*, 1999.

[6] D. Hill and N. Woo. The benefits of flexibility in lookup table-based FPGAs. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 12(2):349–353, 1993.

[7] H.Mrabet, Z.Marrakchi, P.Souillot, and H.Mehrez. Performances improvement of FPGA using novel multilevel hierarchical interconnection structure. *ICCAD, San Jose*, 2006.

[8] S. Kaptanoglu, G. Bakker, A. Kundu, I. Corneillet, and B. Ting. A new high density and very low cost reprogrammable FPGA architecture. *Proceedings of the 1999 ACM/SIGDA seventh international symposium on Field programmable gate arrays*, pages 3–12, 1999.

[9] J. Kouloheris and A. El Gamal. FPGA Area versus Cell Granularity-Lookup Tables and PLA Cells. *FPGA*, 92:9–14, 1992.

[10] B. Landman and R. Russo. On Pin Versus Block Relationship for Partition of Logic Circuits. *IEEE Transactions on Computers*, 20(1469-1479), 1971.

[11] A. Marquardt, V. Betz, and J. Rose. Using cluster based logic blocks and timing-driven packing to improve fpga speed and density. *Proceedings of the International Symposium on Field Programmable Gate Arrays*, pages 39–46, 1999.

[12] Z. Marrakchi, H. Mrabet, E. Amouri, and H. Mehrez. Efficient tree topology for fpga interconnect network. In *ACM Great Lakes Symposium on VLSI*, pages 321–326, 2008.

[13] L. McMurchie and C. Ebeling. Pathfinder: A Negotiation-Based Performance-Driven Router for FPGAs. *Proc.FPGA'95*, 1995.

[14] J. Rose, R. Francis, D. Lewis, and P. Chow. Architecture of field programmable gate arrays: The effect of logic functionality on area efficiency. pages 1217–1225, 1990.

[15] T.Cormen, C.Leiserson, and R.Rivest. Introduction to algorithms. *MIT Press, Cambridge*, 1990.

[16] S. Yang. Logic synthesis and optimization benchmarks, version 3.0, 1991.