Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2009, Article ID 259837, 13 pages
doi:10.1155/2009/259837

Research Article

FPGA Interconnect Topologies Exploration

Zied Marrakchi, Hayder Mrabet, Umer Farooq, and Habib Mehrez

LIP6, Université Pierre et Marie Curie, 4, Place Jussieu, 75252 Paris, France

Correspondence should be addressed to Zied Marrakchi, zied.marrakchi@lip6.fr

Received 1 December 2008; Accepted 20 July 2009

Recommended by J. Manuel Moreno

This paper presents an improved interconnect network for Tree-based FPGA architecture that unifies two unidirectional programmable networks. New tools are developed to place and route the largest benchmark circuits, where different optimization techniques are used to get an optimized architecture. The effect of variation in LUT and cluster size on the area, performance, and power of the Tree-based architecture is analyzed. Experimental results show that an architecture with LUT size 4 and arity size 4 is the most efficient in terms of area and static power dissipation, whereas the architectures with higher LUT and cluster size are efficient in terms of performance. We also show that unifying a Mesh with this Tree topology leads to an architecture which has good layout scalability and better interconnect efficiency compared to VPR-style Mesh.

1. Introduction

The work presented in this paper can be divided into two parts. In the first part, we present an improved Tree-based FPGA architecture. In the second part of the paper, the architecture presented in the first part is used for the improvement of connection blocks and intracluster interconnect topologies in a cluster-based Mesh FPGA architecture.

The motivation behind the work presented in the first part is to reduce the domination of the interconnect area in field programmable arrays (FPGAs). In FPGAs interconnect area takes up to 90% of the total area while the remaining 10% is used by the logic part of the architecture. Domination by interconnect greatly affects the delay and area efficiency of the architecture. In [1] the authors have shown that the best way to improve circuit density is to balance logic blocks and interconnect utilization. In this paper we present an improved Tree-based FPGA (MFPGA) architecture where interconnect and logic utilizations are controlled using different architectural parameters, and it is shown that by reducing the logic occupancy of the architecture, we can increase the interconnect utilization of the architecture resulting in over all area reduction. Also, in this part we investigate the effect of LUT and cluster size on the area, performance, and power dissipation of a Tree-based FPGA. Many studies in the past several years were carried out to see the effect of LUT and cluster size on the density and performance of FPGA architecture [2–5]. But all the work previously done in this context focuses on the Mesh-based FPGA architecture, and no work has been done for Tree-based architectures up to now.

The motivation behind the second part of the paper is the optimization of connection blocks and intracluster interconnect topologies in a cluster-based Mesh FPGA architecture. There are different ways to connect signals to the LUT input muxes. In Xilinx Virtex architectures [6], the routing tracks are connected directly to the input muxes. In the VPR [7] and the Altera Stratix [8] architectures, the routing tracks are connected to the input muxes via an intermediate level of muxes called connection block. VPR-style interconnect has a sparsely populated connection block and a fully populated intracluster crossbar. The fully populated intracluster crossbar is simple but takes no advantage of the logical equivalence of LUT inputs and causes a significant inefficiency. Lemieux and Lewis [9] improved the basic VPR-style interconnect in two ways. They proposed an approach to generate highly routable optimized connection block. Furthermore, they showed that the intracluster full crossbar can be depopulated to achieve significant area reduction without performance degradation. A practical example is Stratix, which depopulates this crossbar by 50% [8]. All

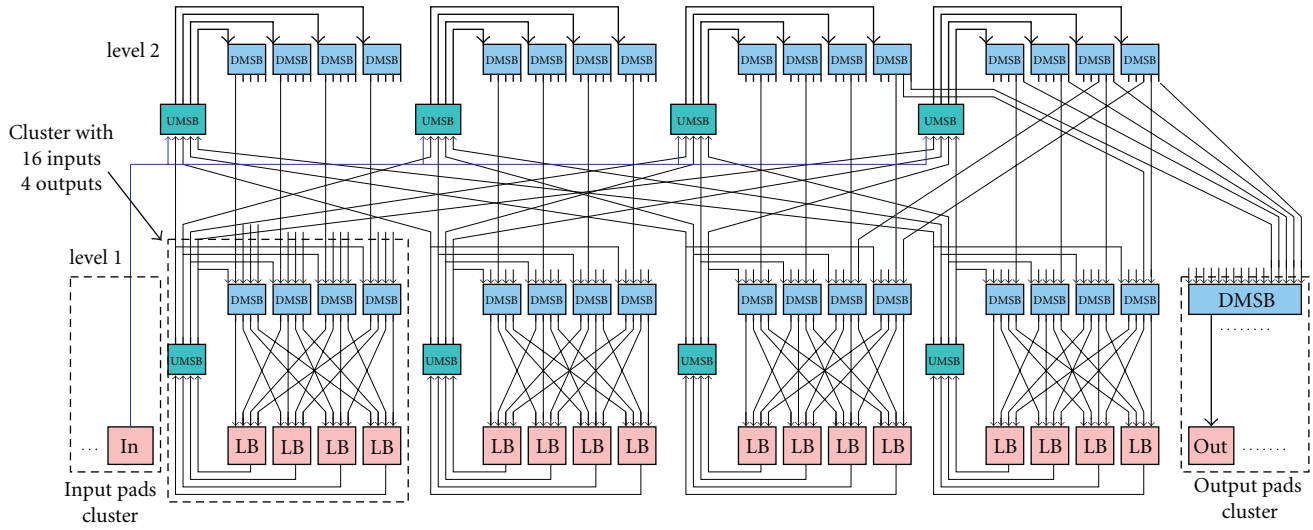


Figure 1: Tree-based interconnect: upward and downward networks.

these studies consider the connection block interconnect level and the intracluster crossbar separately. In [10], authors investigated joint optimization of both crossbars and proposed a new class of efficient topology. Nevertheless in the intracluster crossbar they optimized only the part connecting external signals to LBs inputs. Using a full crossbar to connect feedbacks (LBs outputs) to LBs inputs is very penalizing and imposes a very low bound on the cluster LBs number. For example, we assume that we have a cluster with 256 LBs and we use a full crossbar to connect feedbacks to 4-Lut inputs. This means that we need 256×1024 switches to route clusters internal signals only, which is very expensive. In this part, our first contribution corresponds to a joint optimization of connection blocks and intracluster interconnect topologies. We optimize both crossbars: (1) connecting external signals to LB inputs (2) connecting feedbacks to LB inputs. Our second contribution consists in using only single-driver interconnect based on unidirectional wires. As illustrated in [11], single-driver interconnect has a good impact on density improvement.

The remainder of the paper is organized as follows. Section 2 describes the Tree-based architecture. In Section 3 we propose suitable techniques to place and route netlists on the Tree-based architecture. In the following section, we present the effect of LUT and cluster size on Tree-based FPGA, then we evaluate architecture routability, and we compare it with the common VPR-Style Mesh architecture. Finally we present the cluster-based Mesh FPGA architecture and then we conclude this paper.

2. Tree-Based Interconnect

We propose a Tree-based architecture called MFPGA (Multilevel FPGA) where LBs (Logic Blocks) are grouped into clusters located at different levels. Each cluster contains a switch block to connect local LBs. A switch block is divided into MSBs (Miniswitch Blocks).

2.1. Interconnect Networks. This architecture unifies two unidirectional networks. The downward network uses a "Butterfly Fat Tree" topology to connect DMSBs (Downward MSBs) to LBs inputs. As shown in Figure 1, the number of DMSBs of a cluster located at level ℓ is equal to the number of inputs of a cluster located at level $\ell - 1$. The upward network connects LBs outputs to the DMSBs at each level. As shown in Figure 1, we use UMSBs (Upward MSBs) to allow LBs outputs to reach a large number of DMSBs and to reduce fanout on feedback lines. The number of UMSBs of a cluster located at level ℓ is equal to the number of outputs of a cluster located at level $\ell - 1$. UMSBs are organized in a way allowing LBs belonging to the same "owner cluster" to reach exactly the same set of DMSBs at each level. Thus positions, inside the same cluster, are equivalent, and LBs can negotiate with their siblings the use of a larger number of DMSBs depending on their fanout.

As shown in Figure 1, input and output pads are grouped into specific clusters and are connected to UMSBs and DMSBs, respectively. Thus, input pads can reach all LBs of the architecture, and output pads can also be reached by all the from different paths.

Using UMSBs and DMSBs greatly enhances routability, but it increases the interconnect switches number. However this increase is compensated by reducing in/out signals bandwidth of clusters at every level. In fact, netlists implemented on FPGA architecture often communicate locally (intraclusters) and this fact can be exploited to reduce the bandwidth of signals with inter-clusters communication. A good estimation of netlists communication locality is given by Rent's Rule [12]. Based on this estimation authors in [13] showed that most netlist Rent's parameters range between 0.5 and 0.65.

2.2. Architecture Rent's Parameter. We define Rent's parameter for an architecture as follows:

$$\mathrm{IO} = c \cdot m^{\ell \cdot p}. \tag{1}$$

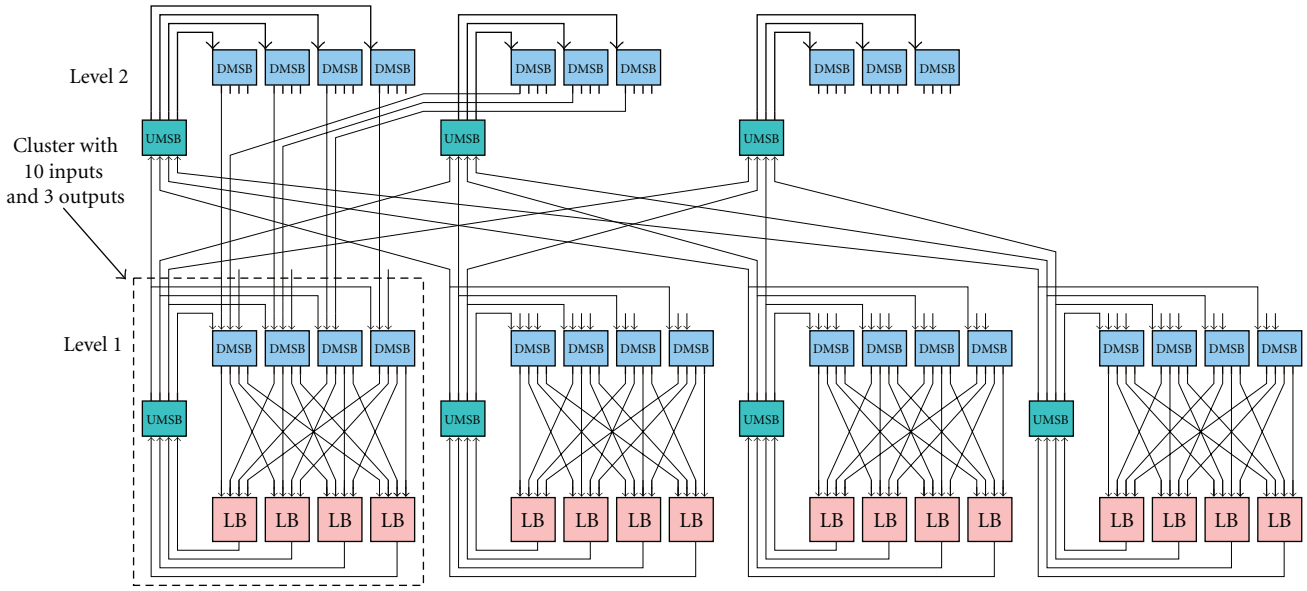


Figure 2: Tree-based interconnect depopulation using Rent's rule (level 1 with $p = 0.73$).

In this formula, ℓ is a Tree level, m is the cluster arity, c is the number of in/out pins of an LB, and IO is the number of in/out pins of a cluster located at level ℓ.

Intuitively, p represents the locality in interconnect requirements. If most of the connections are routed locally and only a few of them communicate to the exterior of a local region, p will be small. In Tree-based architecture, both the upward and downward interconnects populations depend on this parameter. As shown in Figure 2, we can depopulate the routing interconnect by reducing the number of inputs of each cluster of level 1 from 16 to 10 and outputs from 4 to 3 ($p = 0.73$). In this case, if we consider an architecture with 2 levels of hierarchy, we get a reduction in interconnect switches number from 521 to 368 (28%). However, a reduction in the value of p reduces the routability of the architecture too. Thus we must find the best tradeoff between interconnect population and logic blocks occupancy. As shown in [1], the best way to improve circuit density is to balance logic blocks and interconnect utilization. So in MFPGA architecture, interconnect occupancy is controlled by p and logic occupancy factor is controlled by N (the leaves (LBs) number in the Tree).

3. Configuration Flow

The way LBs are distributed between Tree clusters has an important impact on congestion. It is worthwhile to reduce external communications, since local connections are cheaper, not only in terms of delay but also in terms of routability, as this allows to get more levels (more paths) for connecting sources to destinations. Another way to decrease congestion consists in eliminating competition between sources to reach their sinks. This can be achieved by depopulating clusters based on netlist instances fanout. Instances with high fanout need more resources to reach their sinks. Thus in the partitioning phase, instances weights are attributed according to their fanout size. We use a top-down recursive partitioning approach. First, we construct the top level clusters, then every cluster is partitioned into subclusters, until the bottom of the hierarchy is reached. Since logic block positions inside the owner cluster are equivalent, the detailed placement phase (Arrangement inside clusters) is done randomly.

After placement, the routing process is started. Interconnect resources are presented by a routing graph with nodes corresponding to wires and LBs pins and edges to switches. We use the negotiation-based algorithm *Pathfinder* to route netlists signals [14].

4. Experimental Evaluation

To evaluate the proposed architecture performance, we place and route the largest MCNC benchmark circuits available and consider as a reference the optimized clustered Mesh (VPR-style) architecture. We use t-vpack [7] to construct clusters and the channel minimizing router VPR 4.3 [7] to route signals. VPR determines the optimal size as well as the optimal channel width W to place and route each benchmark circuit. In [15], author showed that the wiring structure in the fat-tree containing N logic blocks is sufficiently regular to permit a layout in $O(N)$ area (the area dictated by the nodes and switches) using $O(\log(N))$ wiring layers. Thus, in our area model we do not consider area dictated by wires. We estimate the layout area as the sum of areas required for all logic cells in an FPGA. As presented in Figure 3, switching cells depend on the interconnect structure and especially on wires directions (unidirectional/bidirectional). We use symbolic standard cells library [16] to estimate the FPGA required area. Different cells areas are presented in Table 1. We use bidirectional wires in the case of Mesh and unidirectional wires for MFPGA.

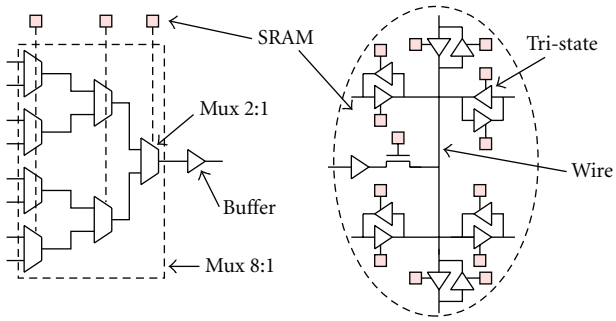


Figure 3: Unidirectional versus bidirectional wires.

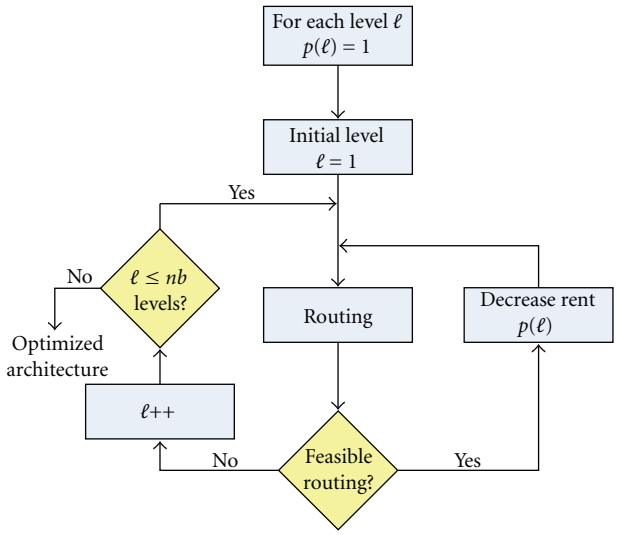


Figure 4: MFPGA architecture optimization flow (bottom-up approach).

Table 1: Standard cells characteristics.

Cell	Area λ^2
Sram	30×50
Tri-state	35×50
Buffer	20×50
Flip-flop	90×50
Mux 2 : 1	35×50

Table 2: Levels Rent's rule parameters (21 benchmark average).

Level	Circuits partitioning	Archi. top-down	Archi. bottom-up	Archi. random
1	0.64	0.98	0.79	0.88
2	0.55	0.88	0.74	0.79
3	0.50	0.80	0.77	0.76
4	0.49	0.75	0.86	0.73
5	0.45	0.59	0.87	0.7

4.1. Architecture Optimization. As explained in Section 2, MFPGA routability and switches number depend on 2 parameters: p (architecture Rent's parameter) and N (number of LBs in the architecture which defines occupancy ratio). To find the best tradeoff between device routability and switches requirement (area) we study MFPGA architectures with various N and p parameters. The purpose is to find for each netlist, the architecture with the smallest area that can implement it. N depends on Tree levels number and clusters arity. For a specific clusters arity, we determine the smallest levels number to implement the circuit. With our tools we can consider, in the same architecture, levels with different p values. Clusters located at the same level have the same Rent's parameter. In the case of Mesh, VPR adjusts the channel width W, and for Tree-based interconnect, we adjust Rent's parameters at every level in order to obtain the smallest architecture.

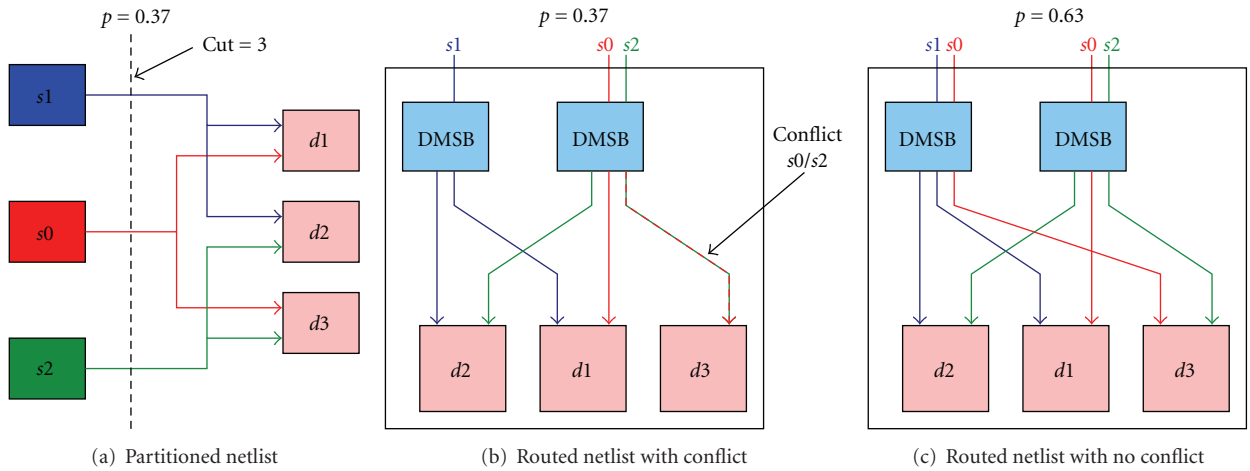
Just like VPR which applies a binary search to find the smallest value of channel width for Mesh architecture, we apply a binary search to determine the smallest value of Rent's parameters for each level of Tree-based architecture. Depending on levels order processing, we tested 3 different approaches.

(i) *Bottom-Up Approach.* As shown in Figure 4, we start by optimizing the lowest level up to the highest one. At each level we apply a binary search to determine the smallest number of input/output signals allowing to route the benchmark circuit.

(ii) *Top-Down Approach.* In this approach we start by optimizing the highest level down to the lowest one. In each level we apply a binary search to determine the smallest number of input/output signals allowing to route the benchmark circuit.

(iii) *Random Approach.* In this approach all levels are optimized in a random fashion. We choose a level randomly and we modify its input/output signals number depending on the previous result obtained at this level, then we move to another one. In this way we move randomly from one level to another until all levels are optimized.

The 3 approaches have the same objective and aim at reducing clusters signals bandwidth at each level. The difference is the order in which levels are processed. In Table 2, we show architecture Rent's parameter (at each level) obtained with each technique. The first column of the table shows Rent's parameters obtained after circuits partitioning. Results correspond to averages of all the 21 circuits. We notice that in all cases, architecture Rent's parameters are larger than partitioned circuits Rent's parameters. This is due to the depopulated switch boxes topology. In fact, to solve routing conflicts, a signal may enter from 2 different DMSBs to reach 2 different destinations located at the same cluster. In Figure 5 we show an example of a partitioned netlist to place and route onto an architecture with LBs inputs number equal to 2 (2 DMSBs in each cluster located at level 1) and clusters arity equal to 3. As shown in the figure if every signal comes from only one DMSB, we cannot solve conflicts. To deal with such a problem we propose to enter the signal driven by $S0$ from 2 different DMSBs. Thus, the



(a) Partitioned netlist (b) Routed netlist with conflict (c) Routed netlist with no conflict

FIGURE 5: A netlist routing example showing architecture Rent's parameter increase.

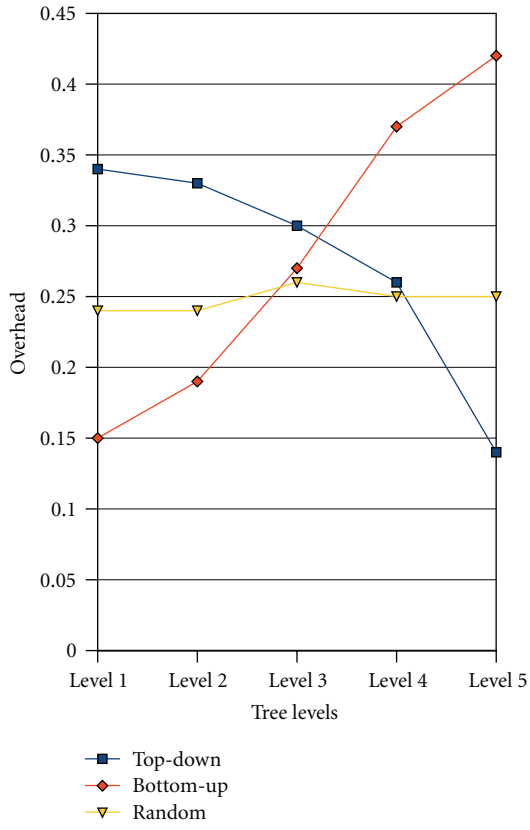


FIGURE 6: Overhead between Architecture and partitioned netlist Rent's parameters (21 benchmark average).

resulting architecture cluster degree is equal to 4, whereas the corresponding partition degree is equal to 3 (number of crossing signals).

In Figure 6, we show the average discrepancy between partitioning and architecture Rent's parameters with each optimizing approach. We notice that in the case of the top-down (bottom-up) approach, overhead increases when we go down (up) in the Tree. This was expected since the top-down (bottom-up) approach first optimizes high (low) levels. With the random approach, we notice that levels overheads are balanced.

TABLE 3: Area and performance comparison between various optimizing approaches (21 benchmark average).

Optimizing approach	Area (λ^2) $\times 10^6$	Critical path switches
Top-down	1498	98
Bottom-up	1326	106
Random	1221	101

A comparison of the average results obtained from the 3 optimizing approaches is shown in Table 3. We notice that with the random approach we obtain the smallest area. This means that optimizing levels randomly allows to avoid local minima and helps to obtain a balanced congestion distribution over levels. The bottom-up approach provides a smaller area than the top-down one, but it is penalizing in terms of critical path switches number. In fact, starting by optimizing low levels means that local routing resources are intensively reduced and signals are routed with resources located at higher levels. Consequently, signals routing uses more switches in series.

To reduce the gap between circuit and architecture Rent's parameters, we must improve the partitioning tool (especially the objective function) to reduce congestion and resources (clusters inputs) required to route signals.

4.2. Clusters Arity and LUT Size Effect. In this section we evaluate the effect of LUTs size k (number of LUT inputs) and cluster arity on area, performance, and static power of MFPGA architecture. In order to evaluate this effect, we performed a series of experiments where LUT size ranges from 3 to 7 and cluster arity ranges from 4 to 8 for each benchmark. Thus we have results for a set of 25 different architectures for each benchmark. First, as shown in Figure 7,

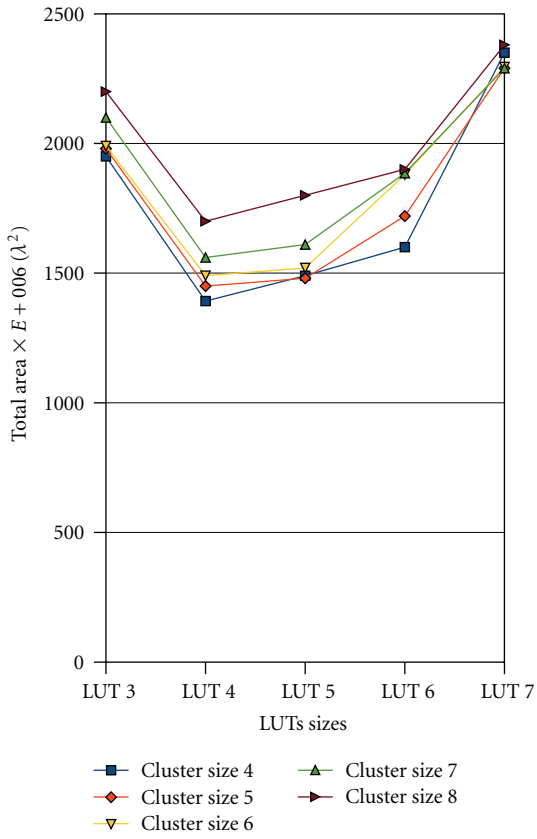


Figure 7: Total area for clusters sizes 4–8 (21 benchmark average).

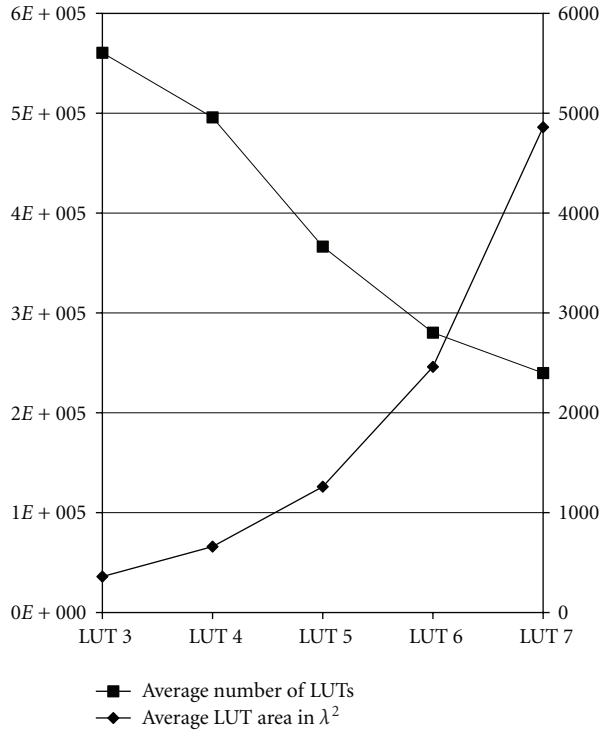


Figure 8: LUTs number and LUT area versus LUT size (for cluster arity = 4).

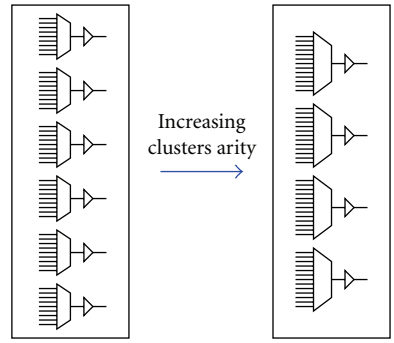


Figure 9: Varying clusters arity ⇒ varying multiplexers sizes and number.

we evaluate the effect of LUT size and cluster arity on MFPGA area. Results correspond to the average area of the 21 largest circuits. We notice that initially there is a reduction in area between $k = 3$ and $k = 4$, but afterwards there is an increase in area with the rise in LUT and cluster sizes. It can be noted from the figure that for the same LUT size, higher cluster arities give higher area values. This is due to the fact that with an increase in the cluster size, there is a decrease in the number of clusters required to implement the circuit but at the same time there is an increase in the area per cluster due to increased value of inputs/outputs bandwidth. In addition, when clusters arity increases, the required multiplexers number decreases but their size grows larger (see Figure 9) and consequently the bound on area efficiency goes down. Hence, these effects combine together and result in higher area values for same LUT size but with higher arity size.

In order to analyze further LUTs size effect on area we divided it into two parts, logic blocks area and interconnect area. From our experimentation we notice that logic area increases with LUT size. This area is the product of the total number of LUTs times the area per LUT. A plot of these two components for clusters arity equal to 4 is shown in Figure 8 (the left vertical axis presents area per LUT in (λ^2) and the right vertical axis presents LUTs number). The logic block area grows exponentially with LUT size as there are 2^k bits in a k-input LUT. Though there is a decline in the number of LUTs with an increase in k (because each LUT can implement more logic functions), the rate of increase in area is steeper than the rate of decrease in LUTs number. Concerning the interconnect area, we notice that it decreases when LUT size increases. Since the rise in logic area is steeper than the decline in interconnect area, we obtain the upward trend in Figure 7.

The second key metric is the critical path delay. Since we have no accurate wire length estimation (we do not have a complete layout generator yet), we only evaluate the number of switches crossed by the critical path. Figure 10 shows the average critical path switches number across the 21 circuits as a function of clusters arities and LUTs sizes. Observing the figure, it is clear that increasing clusters arity and LUTs size decreases the number of switches crossed by the critical path. This behavior is due to the decrease of the number of

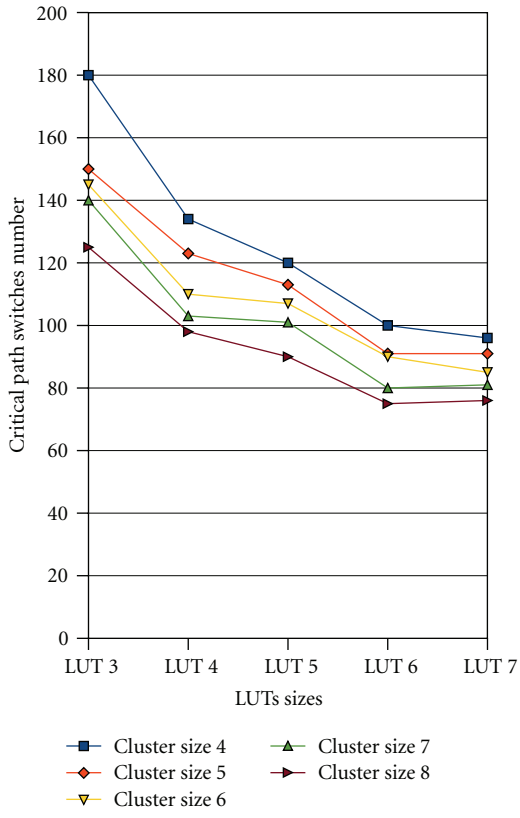


Figure 10: Critical path switches number clusters sizes 4–8 (21 benchmark average).

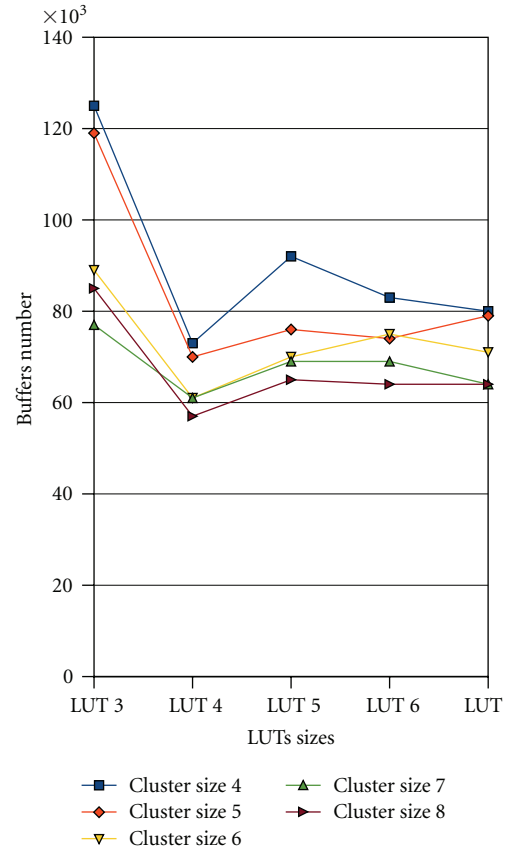


Figure 11: Buffers number clusters sizes 4–8 (21 benchmark average).

Table 4: Levels Rent's parameters for 2 circuits.

Circuits	Level 1	Level 2	Level 3	Level 4	Level 5
apex2	1	0.89	0.86	0.84	0.77
tseng	0.79	0.79	0.79	0.72	0.67

LUTs and clusters in series on the critical path. Nevertheless, to get an idea about the accurate delay we have to consider the increase of intrinsic LUT delay when its size increases.

According to [17], buffers and SRAM are the major factors behind static power dissipation. Therefore, in order to get an idea about LUT and cluster size effect on static power dissipation, we evaluate buffers and SRAM cells numbers as a function of LUT and cluster size. We assume that we insert a buffer at the output of every multiplexer. We notice, as shown in Figure 11, that initially there is a reduction in buffers number when 4-LUTs are used instead of 3-LUTs, and afterwards buffers number increases with LUT size. Also it can be noticed from the Figure 11 that for the same LUT size, higher cluster sizes have a smaller number of buffers. This is due to the fact that when we increase cluster arity, multiplexers sizes increase and their number decreases and consequently buffer number also decreases (see Figure 9). As shown in Figure 12, we notice that SRAM points number decreases when LUT size increases from 3 to 4 and increases afterwards. Clearly, results for all clusters sizes show consistently that LUT size 4 gives minimum leakage energy compared to other LUT sizes. This result is expected since LUT size 4 achieves the highest total-area efficiency.

To get a good tradeoff between area and path delays reduction, using different LUTs sizes is necessary. This was confirmed by the Stratix II architecture [18] where authors showed that the use of ALMs (Adaptive Logic Blocks) gives a good tradeoff between area and critical path delay reductions.

4.3. Area Efficiency. Here we compare MFPGA to the Mesh-based architecture in terms of area efficiency. In both cases we consider architectures with clusters arity 4 and LUT size 4. In each case, we determine the smallest architecture implementing every benchmark circuit. In the case of Mesh we use VPR to find the smallest channel width, and in the case of MFPGA we use the random optimizing approach described in Section 4.1 to determine the smallest Rent's parameters.

In Figure 13, we observe that the Tree-based architecture has a better density for all the 21 benchmark circuits. On average with the Tree architecture we save 56% of the total area. We achieve a 42% area gain with *alu*4 (smallest circuit

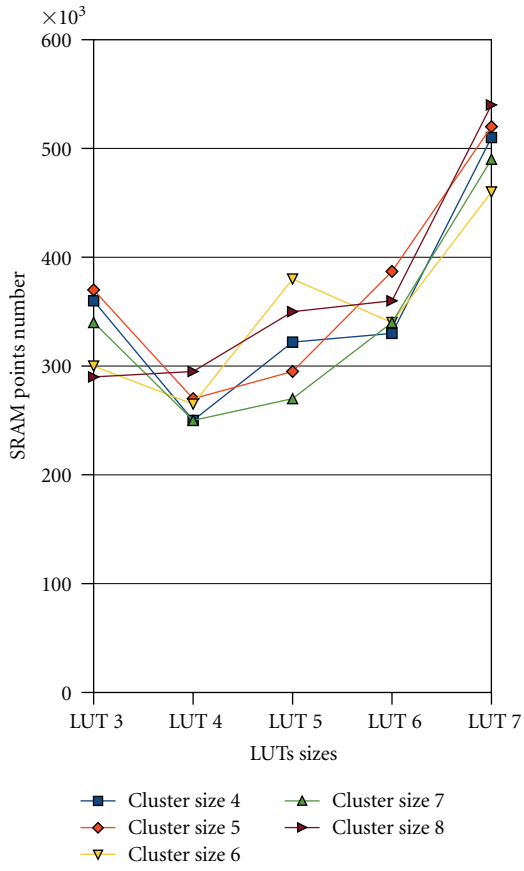


Figure 12: SRAM cells number clusters sizes 4–8 (21 benchmark average)

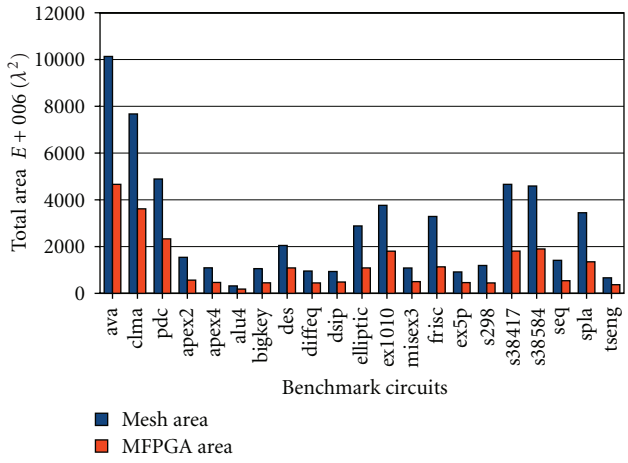


Figure 13: MFPGA area versus Mesh area (21 benchmark circuits).

584 LUTs) and 60% with *ava* (largest circuit 14964 LUTs). This confirms that Tree-based interconnect is very attractive for both small and large circuits.

We compare the areas of both architectures using a refined estimation model of effective circuit area. The Mesh

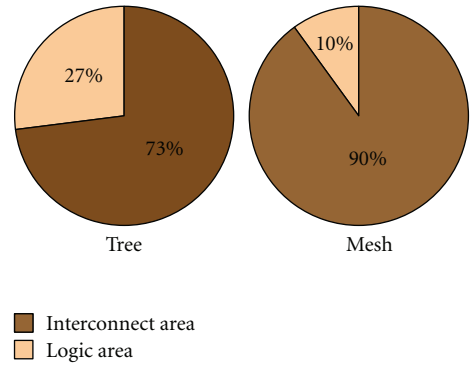


Figure 14: Area distribution between interconnect and logic blocks: Tree and Mesh cases.

area is the sum of its basic cells areas like SRAMs, tri-states, multiplexers and buffers. The same evaluation is made for the Tree, composed of SRAMs, multiplexers, and buffers. Both architectures use the same symbolic cells library.

The Tree architecture efficiency is due essentially to its ability to control simultaneously logic blocks occupancy and interconnect population, based on LBs number N and architecture Rent's parameter p, respectively. For example in the case of apex2 circuit, we use an architecture with high logic occupancy (91%) and high Rent's parameters as shown in Table 4. In the case of tseng circuit, we have a low occupancy (51%) and we achieve routability with a low architecture Rent's parameters as illustrated in Table 4. This confirms that we can balance interconnect and logic blocks utilization with the help of logic occupancy decreasing and congestion spreading. In fact, we use a high-interconnect/low-logic utilization approach which is in direct opposition to the high logic utilization approach that has been adopted for Mesh-based FPGA [7]. As shown in Figure 14, unlike Mesh case where interconnect occupies 90% of the overall area, in Tree-based architecture interconnect occupies 73%. Compared to Mesh architecture, we have a 20% lower occupancy; the extra logic area allows us to exploit interconnect better and to reduce its area by 69%.

5. Unifying Mesh and Tree

We showed that with a Tree-based topology, we obtain good density and we cut area by a factor of 2 compared to Mesh. Nevertheless, based on our layout experimentation we noticed that this Tree-based architecture is penalizing in terms of physical layout generation. It does not support scalability and does not fit with a planar chip structure, especially for large circuits. Conversely the Mesh and in specially the Mesh of Tree [19] has a good physical scalability since it corresponds to an array of repeated tiles. Once a tile layout is generated we can abut it to generate layout of selectable size. In addition, as shown in Figure 15, unlike

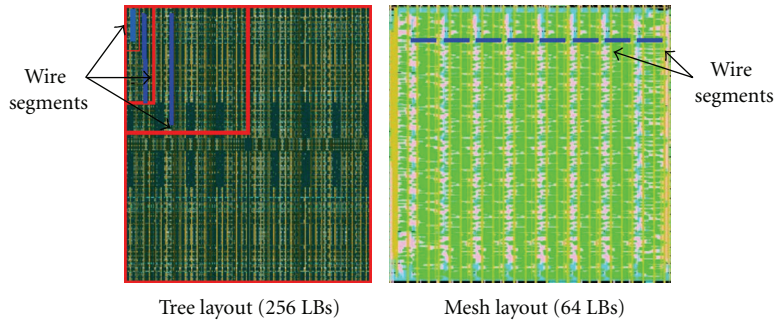


Figure 15: Layout view of Mesh and Tree interconnect structures.

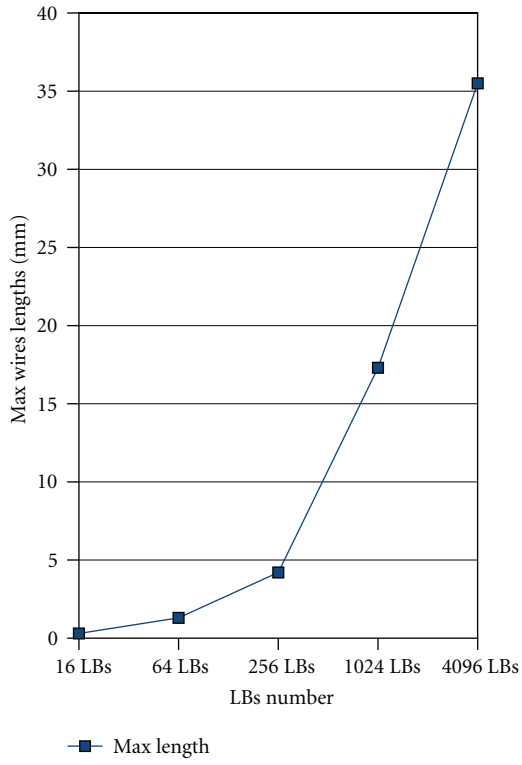


Figure 16: Maximum wire lengths depending on Tree size (arity 4).

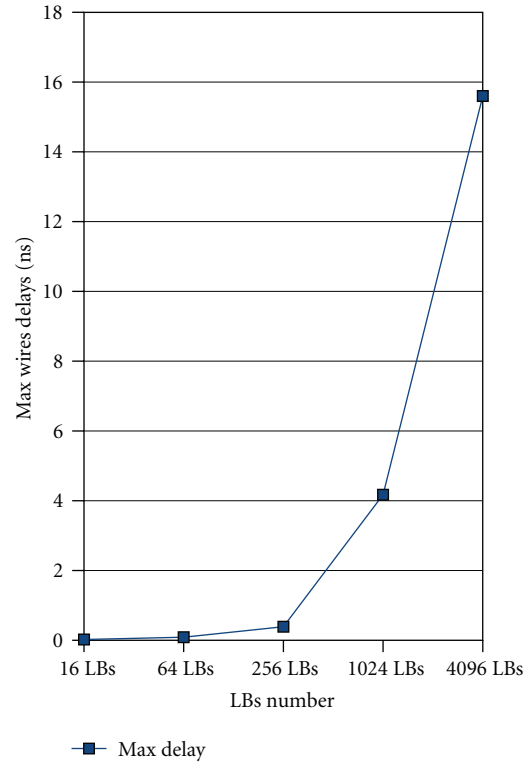


Figure 17: Maximum wire delays depending on Tree size (arity 4).

Mesh where the largest wiring distance is fixed, in the case of a Tree, wiring length increases as we move towards higher levels. An accurate estimation of wire lengths is presented in Figure 16. Maximum wires lengths are evaluated based on layout generated in 130 nm technology. We notice from Figure 17 that in the case of architectures larger than 512 LBs, wires delays become critical and dominant compared to switches delays. It is essential to cut down wire lengths to reduce the quadratic delays growth. For this purpose we propose to use a Mesh interconnect structure from this break even point (512 LBs) to reduce wires delays. In this way wires lengths depend only on Mesh clusters size and no more on total architecture size.

To take advantage of the positive points of both topologies, we propose an architecture where LBs are connected into a cluster (Mesh cluster) with a local interconnect built as a Tree. Mesh clusters are connected with an external interconnect with a Mesh topology. We use the same Tree topology presented previously. In the Mesh interconnect we use only unidirectional wires, since in [11], authors show that single driver interconnect can lead to 25% improvement in area density. Each Mesh cluster is surrounded by four channels which are connected by Switch Boxes (SBs). We do not use connection blocks in the Mesh to connect channel tracks to cluster inputs and outputs. Actually, as presented in [10], interconnect is better optimized when

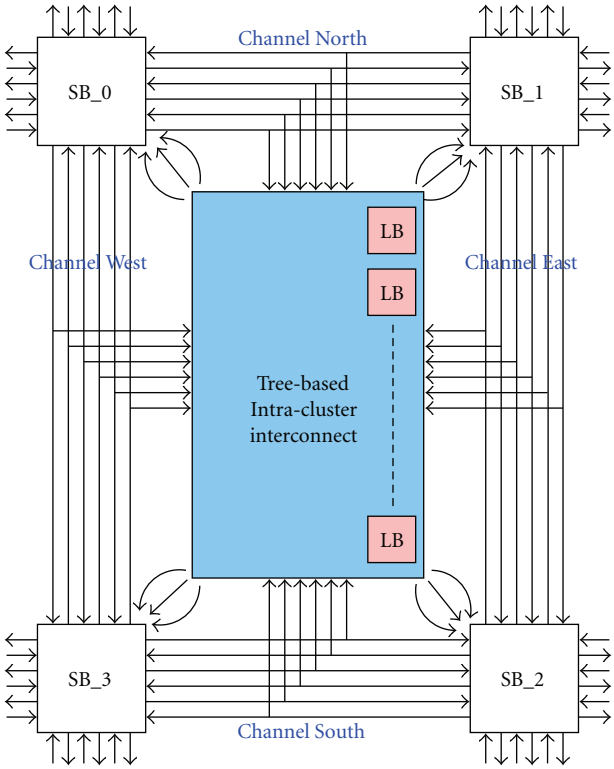


FIGURE 18: Node of Mesh of Tree architecture.

a connection block is combined with the cluster local interconnect.

5.1. Mesh Cluster Interface. As presented in Figure 18, each cluster is connected to the 4 adjacent channel tracks. The cluster input and output connectors are equally distributed on the 4 sides. On all sides we have the same number of inputs and outputs. The distribution of inputs and outputs is illustrated in Figure 19. Input signals (output signals) are grouped together into input Superpins (output Superpins) located at level ℓ_{in} (ℓ_{out}) of the Tree.

(i) Each input Superpin contains 4 inputs connected to the 4 adjacent channels. Each input is connected to all UMSBs located at level $\ell + 1$ of the Tree. In this way the 4 inputs are logically equivalent and can reach all Tree LBs.

(ii) Each output Superpin contains 4 outputs connected to the 4 adjacent switch boxes. Each output is connected to all DMSBs placed at level $\ell + 1$ of the Tree. In this way the 4 outputs are logically equivalent and can be reached from all Tree LBs.

This distribution has an important impact on routability and eliminates constraints in the placement of LBs inside Tree clusters. All 4 Mesh cluster sides have the same number of inputs and outputs. Side inputs and outputs numbers depend on the number of Tree leaves and on the level where they are connected:

$$Nb_{\text{in}} = \frac{N}{k^{\ell_{\text{in}}+1}},$$
$$Nb_{\text{out}} = \frac{N}{k^{\ell_{\text{out}}+1}}, \tag{2}$$

where k is Tree clusters arity. N is the number of Tree leaves. ℓ_{in} and ℓ_{out} are levels where input and output Superpins are located, respectively. As explained previously the Tree-based local interconnect of a cluster can be depopulated using Rent's parameter.

5.2. Mesh Routing Interconnect. As presented in Figure 18, cluster-based Mesh architecture is composed of logic blocks clusters, switch boxes, and in/out pads. Interconnection between clusters is done by routes through switch boxes, along horizontal and vertical routing channels.

In the Mesh interconnect structure we use only single-driver unidirectional wires; in [11], authors show that single-driver-based interconnect leads to a 25% improvement in area density. Each Mesh cluster is surrounded by 4 channels which are connected by Switch Boxes (SBs). As described in Figure 18, Mesh cluster input signals are connected to the 4 adjacent channel tracks. Thus, channel width W is given by

$$W = \frac{Nb_{\text{in}}}{4} = \frac{N}{k^{\ell_{\text{in}}+1}}. \tag{3}$$

A Mesh Switch Box (SB) allows to connect horizontal and vertical channel tracks together and also to cluster outputs. SB inputs come from the 4 channel tracks and from the 4 adjacent clusters outputs. SB outputs are connected to the 4 adjacent horizontal and vertical channels. Since we use a single-driver-based interconnect, each SB output is driven by a multiplexer. Switch boxes have a "disjoint" topology. As presented in Figure 20(b), input track j of channel i is connected to output track j of channel h with $h \neq i$. SBs allow also to connect Mesh cluster outputs to channels tracks. As illustrated in Figure 20(a), each cluster output is connected to all switch box outputs located on the 4 sides.

5.3. Configuration Flow. The configuration flow used to implement benchmark netlists on the proposed architecture is described in Figure 21. First netlist instances are partitioned among the N Mesh clusters. We obtain one external netlist describing Mesh clusters communication and N internal netlists describing instances communication inside each cluster. The external netlist is used to place Mesh clusters on the 2D grid array. All internal netlist instances are partitioned among Tree clusters. We use a multilevel mincut partitioner based on FM refinement heuristics. Once all instances are placed on the Mesh of Tree sites, we run signals routing. Routing consists in assigning netlist signals to routing resources in such a way that no routing resource

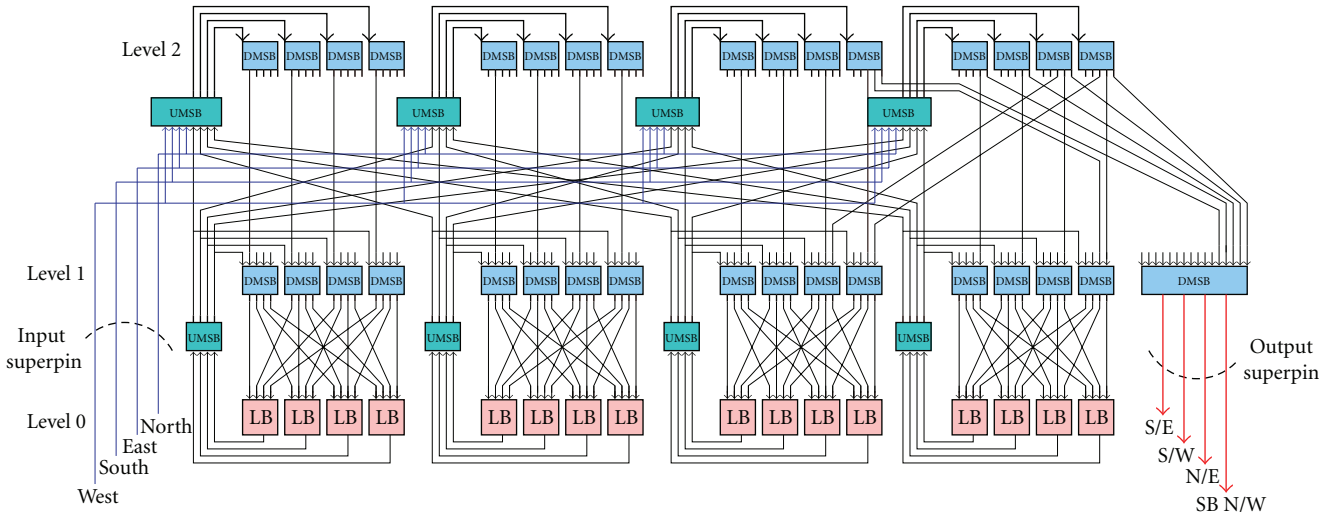
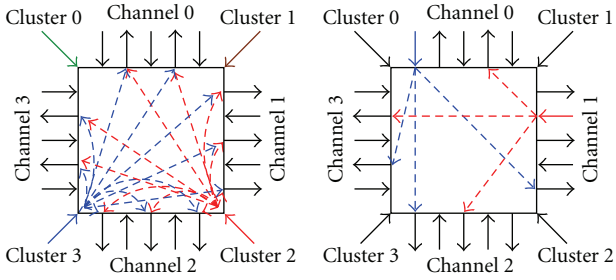


Figure 19: Cluster interface: external input and output connections.



(a) SB: Cluster outputs to channel tracks

(b) SB: channel tracks to channel tracks

Figure 20: Mesh switch box topology.

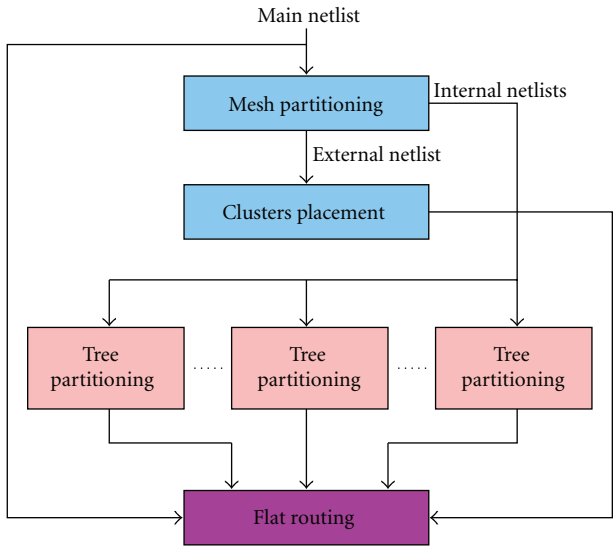


Figure 21: Mesh of Tree configuration flow.

is shared by more than one net. For this purpose we model routing resources by a flat direct graph, where nodes correspond to wires, and LBs pins and edges correspond to switches. We use the *PathFinder* algorithm to route signals on the resulting graph.

5.4. Experimentation. To evaluate the proposed architecture and tool performances, we place and route the 3 largest MCNC benchmark circuits and the *ava* circuit which is the largest design containing only LUTs. We consider as references the optimized cluster-based (VPR-style) Mesh and the MFPGA architectures. We map the 4 largest benchmark circuits on the Mesh of Tree architecture. We consider an architecture with unidirectional wires and Mesh clusters size equal to 256 LBs. Every cluster has 256 inputs and 64 outputs equally distributed on the 4 sides. This is obtained by putting input Superpin at Tree level 0 and output Superpin at level 1. For every benchmark circuit we adjust only the Mesh clusters array size. We do not tailor every Tree interconnect flexibility to every circuit. The Mesh channel width is equal to 64 and Tree signals growth rate p is equal to 0.88. The Mesh of Tree switches requirement and its distribution among Tree and Mesh levels is presented in Figure 22. As shown in Figure 23, we notice that, compared to the VPR-based Mesh architecture, total area is reduced by 42%. This is due essentially to the depopulated intracluster crossbar. In fact with $p = 0.88$ the Tree required switches number is equal to 20×10^3 switches only.

We also notice that, compared to a stand-alone Tree, the total area is increased by 28%. This increase is compensated by the Mesh of Tree layout generation simplicity and wires length reduction, compared to stand-alone Tree, especially when we target large circuits sizes. In this case, wires lengths depend only on Mesh clusters sizes and not on architecture total LBs number.

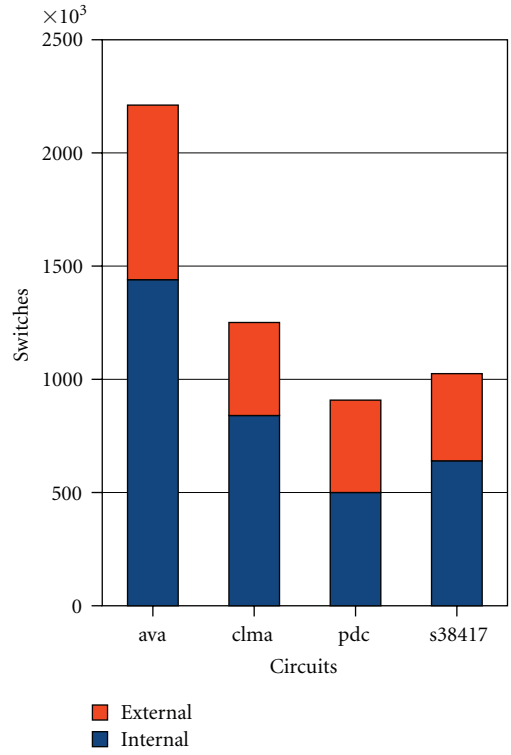


Figure 22: Interconnect distribution in Mesh of Tree architecture.

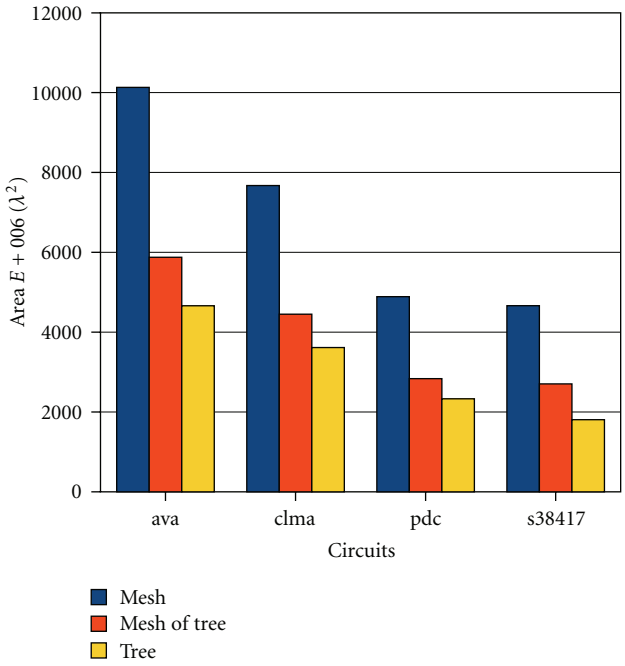


Figure 23: Comparison of various FPGA architectures areas.

6. Conclusion

We proposed a Tree-based architecture with high interconnect and low logic utilizations. Based on the largest MCNC benchmark implementation, we showed that this architecture has better area efficiency than the common VPR-Style clustered Mesh. We showed that in general LUTs with size 4 and cluster size 4 produce most efficient results in terms of area and static power dissipation for Tree-based FPGA. We also determined the evolution of the number of switches crossed by the critical path as a function of LUT and cluster size and we showed 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. Nevertheless, this Tree-based architecture is penalizing in terms of physical layout generation. To deal with such problem we proposed an architecture unifying both Mesh and Tree strong points. The Mesh of Tree has a good physical scalability: once the cluster layout is generated we can abut it to generate Mesh layouts with the desired size and shape factor. The proposed Mesh of Tree architecture is a good tradeoff between area density and layout scalability.

References

[1] A. DeHon, "Balancing interconnect and computation in a reconfigurable computing array (or, why you don't really want 100% LUT utilization)," in *Proceedings of the 7th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '99)*, pp. 69–78, Monterey, Calif, USA, February 1999.

[2] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 288–298, 2004.

[3] J. Rose, R. Francis, D. Lewis, and P. Chow, "Architecture of field-programmable gate arrays: the effect of logic block functionality on area efficiency," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 5, pp. 1217–1225, 1990.

[4] S. Kaptanoglu, G. Bakker, A. Kundu, I. Corneillet, and B. Ting, "A new high density and very low cost reprogrammable FPGA architecture," in *Proceedings of the 7th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '99)*, pp. 3–12, Monterey, Calif, USA, February 1999.

[5] D. Hill and N. Woo, "The benefits of flexibility in lookup table-based FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 2, pp. 349–353, 1993.

[6] Xilinx Corp, http://www.xilinx.com.

[7] V. Betz, A. Marquardt, and J. Rose, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[8] D. Lewis, V. Betz, D. Jefferson, et al., "The Straix™ routing and logic architecture," in *Proceedings of the 11th ACM/SIGDA ACM International Symposium on Field Programmable Gate Arrays (FPGA '03)*, pp. 12–20, Monterey, Calif, USA, February 2003.

[9] G. Lemieux and D. Lewis, *Design of Interconnection Networks for Programmable Logic*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.

[10] W. Feng and S. Kaptanoglu, "Designing efficient input interconnect blocks for LUT clusters using counting and entropy," in *Proceedings of the 15th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '07)*, pp. 23–32, Monterey, Calif, USA, February 2007.

[11] G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and single-driver wires in FPGA interconnect," in *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT '04)*, pp. 41–48, Brisbane, Australia, December 2004.

[12] B. Landman and R. Russo, "On a pin versus block relationship for partitions of logic graphs," *IEEE Transactions on Computers*, vol. 20, no. 12, pp. 1469–1479, 1971.

[13] J. Pistorius and M. Hutton, "Placement rent exponent calculation methods, temporal behaviour and FPGA architecture evaluation," in *Proceedings of the International Workshop on System Level Interconnect Prediction*, pp. 31–38, Monterey, Calif, USA, April 2003.

[14] L. McMurchie and C. Ebeling, "PathFinder: a negotiation-based performance-driven router for FPGAs," in *Proceedings of the 3rd ACM International Symposium on Field-Programmable Gate Arrays (FPGA '95)*, pp. 111–117, Monterey, Calif, USA, February 1995.

[15] A. DeHon, "Compact, multilayer layout for butterfly fat-tree," in *Proceedings of the 12th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '00)*, pp. 206–215, Bar Harbor, Me, USA, July 2000.

[16] A. Greiner and F. Pechêux, "Alliance: a complete set of CAD tools for teaching VLSI design," in *Proceedings of the 3rd EuroChip Workshop*, pp. 230–237, September 1992.

[17] S. Kaptanoglu, "Power and the future FPGA architectures," in *Proceedings of the International Conference on Field Programmable Technology (ICFPT '07)*, pp. 241–244, Kitakyushu, Japan, December 2007.

[18] D. Lewis, E. Ahmed, G. Baeckler, et al., "The Stratix II logic and routing architecture," in *Proceedings of the 13th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '05)*, pp. 14–20, Monterey, Calif, USA, February 2005.

[19] A. DeHon, "Unifing mesh and tree-based programmable interconnect," *IEEE Transactions on VLSI Systems*, vol. 12, no. 10, pp. 1051–1065, 2004.