Efficient Tree Topology for FPGA Interconnect Network

Zied Marrakchi, Hayder Mrabet, Emna Amouri and Habib Mehrez
LIP6, Université Pierre et Marie Curie
4, Place Jussieu, 75005 Paris, France
zied.marrakchi@lip6.fr

ABSTRACT

This paper presents an improved Tree-based architecture that unifies two unidirectional programmable networks: A predictible downward network based on the Butterfly-Fat-Tree topology, and an upward network using hierarchy. Studies based on Rent's Rule show that switch requirements in this architecture grow slower than in traditional Mesh topologies. New tools are developed to place and route several benchmark circuits on this architecture. Experimental results show that the Tree-based architecture can implement MCNC benchmark circuits with an average gain of 54% in total area compared with Mesh architecture.

Categories and Subject Descriptors

B.6.1[**Logic Design**]: Design Styles

General Terms

Design, Experimentation, Performance

Keywords

FPGA, Hierarchy, Interconnect, Rent's rule, Routing

1. INTRODUCTION

Up to 90% of a Field Programmable Gate Array (FPGA) chip is occupied by the programmable interconnect, including wires, switches and configuration bits. Modern Mesh FPGAs use clustering to improve the area and delay efficiency of the routing architecture [3][9]. This shows that partitioning and hierarchy become unavoidable for hardware and software developments. A multilevel hierarchical FPGA (MFPGA) architecture, where logic blocks and routing resources are sparsely partitioned into a multilevel clustered structure, were presented in [7]. Authors proved that this architecture has better density than common VPR-style Mesh architecture [3]. Nevertheless, the Tree based architecture cannot implement highly congested netlists. In this work we propose an improved MFPGA architecture interconnect to get better routability without degrading area efficiency. Next section describes the improved Tree-based architecture (MFPGA) and evaluates its switches requirement growth. In section 3 we propose suitable techniques to place and route netlists on the Tree-based architecture. Finaly, based on the largest MCNC benchmarks implementation, we evaluate architecture routability and we compare its area efficiency to the common VPR-Style Mesh architecture.

GLSVLSI'08, May 4–6 2008, Orlando, Florida, USA.

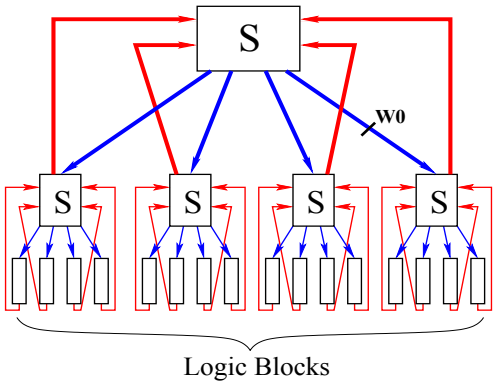

Figure 1: Upward and downward networks

2. TREE-BASED INTERCONNECT

In a previous work [7] a first hierarchical Multilevel FPGA architecture (MFPGA) was designed and experimentally evaluated. As presented in figure 1, this architecture unifies two unidirectional networks. The downward network has a "Butterfly Fat Tree" topology and allows to connect Switch blocks to LBs (leaves) inputs. The upward network uses a limited connectivity Tree to connect LBs outputs to Switch Blocks. While providing good flexibility and some interesting features like an almost predictible routing once the placement is defined, this approach revealed some drawbacks hindering highly congested netlists routing:

- The very depopulated upward network, which only allows each LB output to reach any destination through paths as the number of levels in the hierarchy, is detrimental for highly congested netlists.
- The placement of clusters (or LBs) inside their owner cluster critically controls available routing resources, thus limiting the freedom to re-arrange them and making impossible to construct carry chains in this type of architecture.

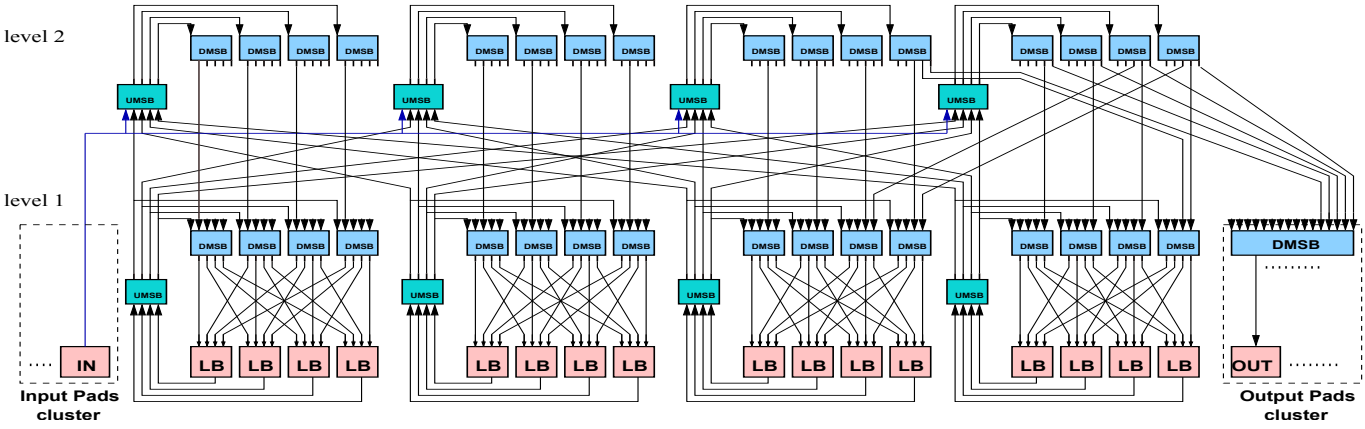

Figure 2: Tree-based interconnect: upward and downward networks

2.1 Interconnect improvement

To alleviate those weaknesses we propose to add routing flexibility by modifying specifically the upward network. We propose, as shown in figure 2, to add Upward Mini Switch Boxes (UMSB). These UMSBs allow LBs outputs to reach a larger number of Downward MSBs (DMSBs). The UMSBs are organized in a way that allows logic blocks (LBs) belonging to the same "owner cluster" (at level 0 or above) to reach exactly the same set of DMSBs at each level. Therefore, we can ensure the following points:

• Pads, clusters or logic blocks positions inside the direct owner cluster become equivalent and we need no more to re-arrange them.

• The interconnect offers more routing paths to connect a net source to a given sink. In this case we are more likely to achieve highly congested netlists routing. In fact, while in the previous architecture each LB output had only a fixed number of reachable DMSBs per level, with the new upward network, LBs can negotiate with their siblings the use of a larger number of DMSBs. This is more efficient for mapping netlists since instances can have different fanout sizes. For example in figure 2, an LB ouput can reach all 4 DMSBs of its owner cluster at level 1 and all the 16 DMSBs of its owner cluster at level 2.

2.2 Interconnect depopulation

When we add UMSBs in the upward network, the number of architecture switches increases. This can be compensated by the reduction of in/out signals bandwidth of clusters in each level. In fact Rent's rule [8] is easily adapted to Tree-based structure:

$$IO = c.k^{\ell.p} \quad (1)$$

Where ℓ is a Tree level, k is the cluster arity, c is the number of in/out pins of an LB and IO the number of in/out pins of a cluster situated at level ℓ.

Intuitively, p represents the locality in interconnect requirements. If most connections are purely local and only few of them come in from 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 3, we can depopulate the routing interconnect by reducing from 16 to 12 the number of inputs in each cluster of level 1 and outputs from 4 to 3 ($p = 0.79$). This induces a reduction from 16 to 12 of the number of DMSBs in each cluster of level 2 and the UMSBs number from 4 to 3. In this case, if we consider an architecture with 2 levels of hierarchy, we get a reduction of the interconnect switches number from 521 to 416 (19%). By doing so the architecture routability is reduced too. Thus we have to find the best tradeoff between interconnect population and logic blocks occupancy. Dehon showed in [5] that the best way to improve circuit density is to balance logic blocks and interconnect utilization. In the proposed architecture, the logic occupancy factor is controled by N, the leaves (LBs) number in the Tree. N is directly related to the number of levels and the clusters arity k.

2.3 Connection with outside

As shown in figure 2, output and input pads are grouped into specific clusters. The cluster size and the level where they are located can be modified to obtain the best design fit. Each input pad is connected to all UMSBs of the upper level. In this way each input pad can reach all LBs of the architecture with different paths.

Similarly, output pads are connected to all DMSBs of the upper level; in this way they can be reached from all LBs through different paths. As one can notice, in/out pads have higher interconnection flexibility than LBs.

2.4 Rent's Rule Based Model

Based on the Rent's rule presented in equation (1), we evaluate the Tree architecture switches requirement.

2.4.1 The Downward Network Model

Clusters situated at level ℓ contain $N_{in}(\ell - 1)$ DMSB with k outputs and $\frac{N_{in}(\ell)+kN_{out}(\ell-1)}{N_{in}(\ell-1)}$ inputs. If we assume that the DMSB are full crossbar devices, we get $k(N_{in}(\ell) + kN_{out}(\ell - 1))$ switches in the switch box of a level ℓ cluster. Since we have $\frac{N}{k^\ell}$ clusters in level ℓ, we get a total number of switches, related to the downward network, given by:

$$\sum_{\ell=1}^{\log_k(N)} k \times N \times \frac{N_{in}(\ell) + kN_{out}(\ell - 1)}{k^\ell}$$

$N_{out}(0) = c_{out}$ is the number of outputs of a Basic Logic Block. Following equation (1), we get $N_{in}(\ell) = c_{in}.k^{\ell.p}$ and $N_{out}(\ell - 1) = c_{out}.k^{(\ell-1)p}$.

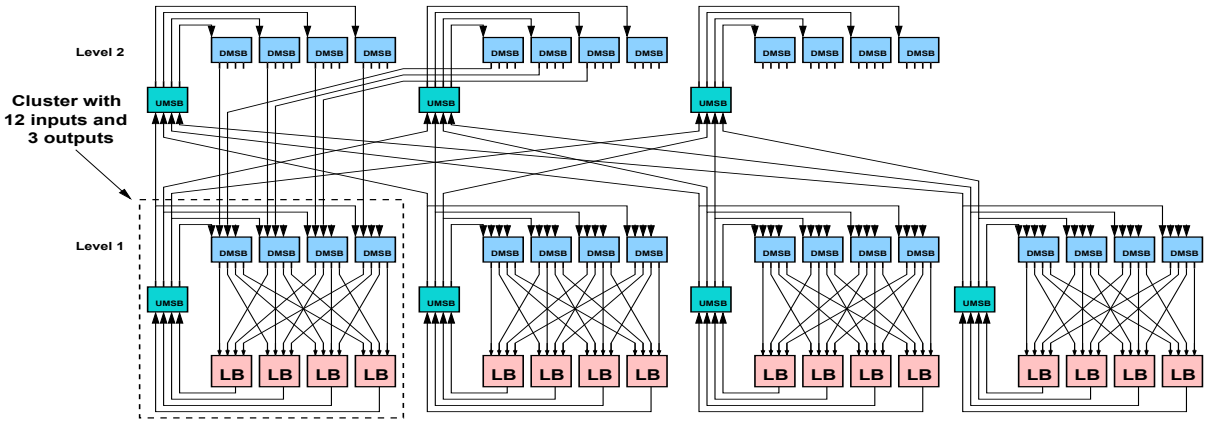

Figure 3: Tree-based interconnect depopulation based on Rent's parameter (level 0 with $p = 0.79$)

The total number of switches used in the downward interconnect is:

$$N_{switch}(down) = N \times (k^p c_{in} + k c_{out}) \times \sum_{\ell=1}^{\log_k(N)} k^{(p-1)(\ell-1)}$$

2.4.2 The Upward Network Model

Clusters situated at level ℓ contain $N_{out}(\ell-1)$ UMSB with k inputs and k outputs. If we assume that the UMSB are full crossbar devices, we get $k^2 \times N_{out}(\ell-1)$ switches in the switch box of a level ℓ cluster. As we have $\frac{N}{k^\ell}$ clusters in level ℓ we get the total number of switches, related to the upward network:

$$\sum_{\ell=1}^{\log_k(N)} \frac{k^2 \times N}{k^\ell} \times N_{out}(\ell-1)$$

$N_{out}(0) = c_{out}$ is the number of outputs of a Basic Logic Block. Following (1), we get $N_{out}(\ell-1) = c_{out}.k^{(\ell-1)p}$. The total number of switches used in the upward interconnect is:

$$N_{switch}(up) = N \times k \times c_{out} \times \sum_{\ell=1}^{\log_k(N)} k^{(p-1)(\ell-1)}$$

The total number of Tree-based interconnect switches is

$$N_{switch}(Tree) = N_{switch}(down) + N_{switch}(up)$$

$$N_{switch}(Tree) = N \times (k^p c_{in} + 2k c_{out}) \times \sum_{\ell=1}^{\log_k(N)} k^{(p-1)(\ell-1)}$$

The number of switches per Logic Block is:

$$N_{switch}(LB) = (k^p c_{in} + 2k c_{out}) \times \sum_{\ell=1}^{\log_k(N)} k^{(p-1)(\ell-1)}$$

$$N_{switch}(LB) = \begin{cases} (k^p c_{in} + 2k c_{out}) \times \frac{1-N^{p-1}}{1-k^{p-1}} & \text{if } p \neq 1 \\ (k^p c_{in} + 2k c_{out}) \times \log_k(N) & \text{if } p = 1 \end{cases}$$

$$N_{switch}(LB) = \begin{cases} O(1) & \text{if } p < 1 \\ O(\log_k(N)) & \text{if } p = 1 \end{cases} \quad (2)$$

2.4.3 Comparison with Mesh Model

Concerning switches per logic block growth, it was established in [5] that in the Mesh architecture:

$$N_{switch}(LB) = O(N^{p-0.5}) \quad (3)$$

Equations (2) and (3) show that in the Tree-based architecture, switches requirement grow more slowly than in common Mesh architecture. These results are encouraging for constructing very large structures, especially when p is less than 1. But this does not mean that our Tree-based topology is more efficient than other architectures, since they do not have the same routability. The best way to check this point is through experimental work, in order to compare area results, using Tree-based and the VPR clustered Mesh FPGA.

3. CONFIGURATION FLOW

To explore the modified architecture we have to adapt the configuration flow. Since logic blocks positions inside the owner cluster are equivalent, the detailed placement phase (Arrangement inside clusters) is eliminated.

3.1 Multilevel partitioning

The way how logic LBs are distributed between Tree clusters has an important impact on congestion. It is worthwhile to reduce external communications, since local connections are cheaper in terms of delay, but also in terms of routability, as it allows to get more levels (more paths) for connecting sources to destinations. Another way to decrease congestion consists in eliminating competition between nets sources reaching 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 which gives priority to global connectivity information. First, we construct the top level clusters, then each cluster is partitioned into sub-clusters, until the bottom of the hierarchy is reached. To perform partitioning we used hMetis [6] since it generates a good solution in a short time thanks to its multi-phase refinement approach.

3.2 Routing

Once the netlist is partitioned into a tree of nested clusters, we attribute randomly to each cluster a position inside its owner (no detailed placement is required). The routing problem consists in assigning the nets that connect placed logic blocks to routing resources in the interconnect structure. The new topology of the upward interconnect adds extra paths to connect a LB to a destination but eliminates the predictability property. Hence we must model the routing resources as a directed graph abstraction $G(V, E)$. The set of vertices V represents the in/out pins of logic blocks and the routing wires in the interconnect structure. An edge between two vertices represents a potential connection between the two vertices. The routing algorithm we implemented is "PathFinder" [10], which uses an iterative, negotiation-based approach to successfully route all nets in a netlist. During the first routing iteration, nets are freely routed without paying attention to resource sharing. Two terminal nets are routed using Dijkstra's shortest path algorithm [11], and multi-terminal nets are decomposed into terminal pairs by the Prim's minimum-spanning tree algorithm [11]. 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.

4. EXPERIMENTAL EVALUATION

To evaluate the proposed architecture and tool performances, we place and route the largest MCNC benchmark circuits, and consider as a reference the optimized clustered Mesh (VPR-style) architecture. This reference architecture uses an uniform routing with single-length segments and a disjoint switch block. Each cluster logic block contains four 4-LUTs, 10 inputs and 4 outputs which are distributed over the cluster sides. LUTs pins are connected to cluster pins using a full local crossbar. Connection block population is defined by $F_{c_{in}}$ and and $F_{c_{out}}$ parameters, where $F_{c_{in}}$ is routing channel to cluster input switch density and $F_{c_{out}}$ is cluster output to the routing channel density. $F_{c_{in}} = 0.5$ and $F_{c_{out}} = 0.25$ are chosen to be consistent with previous work [1]. The choice of segments length equal to 1 is due to the fact that, in the proposed Tree architecture, we used only segments of length 1 (no wires are crossing more than one level). We use t-vpack [2] to construct clusters and the channel minimizing router VPR 4.3 [4] to route the Mesh. VPR chooses the optimal size as well as the optimal channel width W to place and route each benchmark circuits. First we evaluate the efficiency of the new Tree-based architecture to implement MCNC benchmark circuits. With the previous MFPGA architecture [7], several of the largest MCNC circuits were unroutable.
As shown in table 1, we achieved all the 20 MCNC largest benchmarks routing. This illustrates the improvement in routing flexibility provided by the new upward network.

4.1 Area Efficiency

We compared the area requirement between Tree architecture and the clustered VPR-style Mesh architecture to implement these benchmarks. As explained in section 2.2, routability and switches number depend on two parameters:

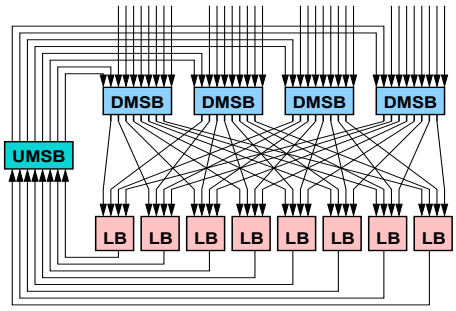

Figure 4: Cluster with arity 8 and $p = 1$

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 (area) requirement we explored Tree-based architectures with various N and p parameters. The purpose was to find for each netlist, the architecture with the smallest area that can implement it. With our tools we can consider, in the same architecture, levels with different p values. This is why in table 1, we present the Rent's average value.
Both Mesh and Tree architectures characteristics are presented in table 1. In the case of Mesh we adjust the channel width W and for the Tree-based interconnect we adjust levels Rent's parameters in order to obtain the architecture which best fits each benchmark.
In table 2, we observe that the Tree architecture has a better density and can implement circuits with lower switches number. An average of 59% reduction of the switches number is achieved. We achieve a 42% switches reduction in the case of the "alu4" smallest circuit and 52% in the case of the "clma" largest circuit. 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 area is the sum of its basic cells areas like SRAMs, Tri-states and Multiplexers. The same evaluation is made for the Tree, composed of SRAMs and Multiplexers. Both architectures use the same cell symbolic library. As presented in table 2, with the Tree we save 54% in the total area compared to Mesh architecture.
The Tree architecture efficiency is due essentially to the ability to control simultaneously the logic blocks occupancy and the interconnect population, based on respectively LBs number N and architecture Rent's parameter p. For example in the case of "apex2" circuit, we used an architecture with a high logic occupancy (91%) and a high Rent's parameter $p = 0.86$. In the case of "alu4" circuit, we have a low occupancy (57%) and we achieve routability with a low architecture Rent's parameter equal to 0.66.

4.2 Clusters Arity Effect

As one can notice, we have considered in table 1 Tree architecture with clusters arity equal to 4. To get an idea about arity effect on architecture density and performances, as shown in table 3, we varied clusters arity and evaluated for each benchmark circuit the required switches and wires number and the resulting critical path. Since we have no information about layout characteristics, we used a basic model based on evaluation of the number of switches crossed

MCNC benchmarks				Clustered Mesh cluster size 4			Tree architecture		
Circuits Names	LUTs Number	IN Pads	OUT Pads	Arch NxN	Occup %	Channel Width	Architecture levels	Occup %	Rent's p
alu4	584	14	8	13x13	86	32	4x4x4x4x4	57	0.66
apex2	1878	39	3	23x23	88	40	4x4x4x4x4x2	91	0.86
apex4	1262	9	19	19x19	87	42	4x4x4x4x4x2	61	0.79
bigkey	1707	263	197	21x21	96	28	4x4x4x4x4x2	83	0.79
clma	8383	61	82	47x47	94	51	4x4x4x4x4x4x4	51	0.77
des	3235	256	245	29x29	96	29	4x4x4x4x4x4	78	0.84
diffeq	1497	64	39	20x20	93	29	4x4x4x4x4x2	73	0.72
dsip	1370	229	197	19x19	95	31	4x4x4x4x4x2	67	0.81
elliptic	3604	131	114	31x31	94	41	4x4x4x4x4x4	87	0.84
ex1010	4589	10	10	35x35	93	43	4x4x4x4x4x4x2	56	0.77
ex5p	1064	8	63	17x17	92	44	4x4x4x4x4x2	51	0.77
frisc	3556	20	116	30x30	98	45	4x4x4x4x4x4	86	0.86
misex3	1397	14	14	20x20	87	36	4x4x4x4x4x2	68	0.84
pdc	4575	16	40	35x35	93	61	4x4x4x4x4x4x2	55	0.79
s298	1931	4	6	23x23	91	27	4x4x4x4x4x2	94	0.72
s38417	6406	29	106	41x41	95	37	4x4x4x4x4x4x2	78	0.70
s38584	6447	39	304	41x41	96	36	4x4x4x4x4x4x2	78	0.75
seq	1750	41	35	22x22	90	40	4x4x4x4x4x2	85	0.84
spla	3690	16	46	31x31	96	53	4x4x4x4x4x4	90	0.93
tseng	1047	52	122	17x17	90	27	4x4x4x4x4x2	51	0.79
Average	2998	82	88		92	38		72.9	0.79

Table 1: Netlists and architectures characteristics

MCNC	Clustered Mesh Cluster size 4			Tree architecture			Gain		
Circuits	SW $\times 10^3$	SRAM $\times 10^3$	Area (λ^2) $\times 10^6$	SW $\times 10^3$	SRAM $\times 10^3$	Area (λ^2) $\times 10^6$	SW %	SRAM %	Area (λ^2) %
alu4	100	74	319	47	43	182	53	41	42
apex2	506	375	1541	173	127	565	65	66	63
apex4	359	267	1092	138	103	466	61	61	57
bigkey	349	253	1056	129	101	450	63	60	57
clma	2541	1879	7672	1031	821	3614	59	56	52
des	667	487	2047	326	247	1087	51	49	46
diffeq	307	226	954	121	108	445	60	52	53
dsip	310	224	934	120	115	500	52	48	46
elliptic	944	701	2883	326	247	1087	65	48	62
ex1010	1234	915	3763	515	410	1804	58	55	52
ex5p	305	224	915	134	103	460	56	54	49
frisc	952	811	3287	346	254	1134	63	68	65
misex3	354	263	1085	163	123	541	53	53	50
pdc	1636	1207	4889	714	523	2329	56	56	52
s298	380	280	1192	121	108	445	68	61	62
s38417	1508	1126	4662	493	439	1807	67	60	61
s38584	1501	1113	4590	535	452	1898	64	59	58
seq	463	343	1411	163	123	541	64	64	61
spla	1144	847	3448	428	299	1350	62	64	60
tseng	216	157	665	126	100	442	41	36	33
Average	788	588	2420	362	280.6	1228.9	59	55	54

Table 2: Tree vs clustered VPR-style Mesh

	Arity 4				Arity 8					Arity 16				
Circuits	SW $\times 10^3$	SRAM $\times 10^3$	Wires $\times 10^3$	C-Path SW	Arch levels	SW $\times 10^3$	SRAM $\times 10^3$	Wires $\times 10^3$	C-Path SW	Arch levels	SW $\times 10^3$	SRAM $\times 10^3$	Wires $\times 10^3$	C-Path SW
apex2	173	127	43	86	8x8x8x4	198	120	27	62	16x16x16	291	114	22	48
apex4	138	103	35	82	8x8x8x4	204	111	28	54	16x16x8	232	103	19	40
clma	1031	821	267	152	8x8x8x8x4	1380	868	199	100	16x16x16x4	1814	854	150	80
dsip	147	115	37	34	8x8x8x4	137	92	20	22	16x16x8	248	104	18	16
ex5p	134	103	34	92	8x8x8x4	174	105	25	64	16x16x8	210	102	18	44
misex3	163	123	40	70	8x8x8x4	165	104	24	58	16x16x8	210	101	18	46
pdc	714	523	175	124	8x8x8x8x2	783	495	110	80	16x16x16x2	907	427	75	64
s38417	493	439	130	108	8x8x8x8x4	645	412	95	72	16x16x16x4	788	393	69	62
Average	491	337.8	126.7	134		604	372.6	85.1	98.4		795.4	352	63.6	77.5

Table 3: Architcture Arity Effect

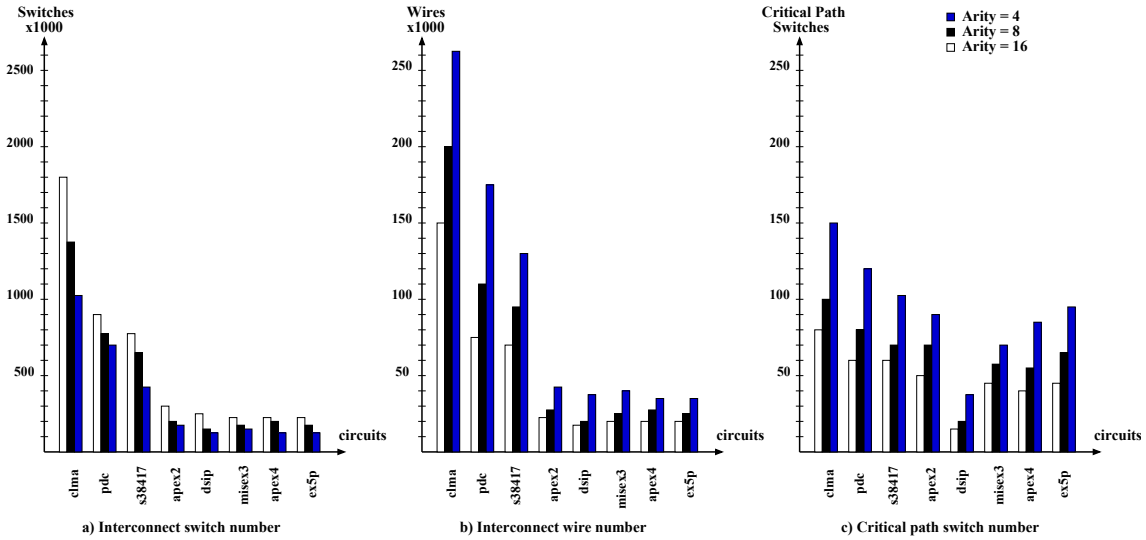

Figure 5: Clusters arity effect

by the critical path to evaluate performances. We notice that when we increase clusters arity, the required switches number increases. When clusters arity increases, the required muxes get bigger and consequently the bound on area efficiency goes down. For example, in the case of architecture with clusters arity 4 we use muxes 4:1 and muxes 5:1. With clusters arity 8, we use muxes 8:1 and muxes 10:1 (figure 4). As shown in figure 5, switches number is increased by 23% when we increase clusters arity from 4 to 8.
When we increase clusters arity, the architecture levels number decraeses. Consequently multiplexers sizes increases and their total number decreases. Thus the total number of wires decreases. For example, as shown in figure 5, wire number is reduced by 32% when we increase clusters arity from 4 to 8.
In terms of performance we notice that the number of switches crossed by the critical path decreases when we increase arity. With larger clusters arity, we can absorb more nets and communication becomes local. For example when we increase clusters arity from 4 to 8, the crossed switches number in the critical path is reduced by 27%.

5. CONCLUSION

The improved Tree-based architecture significantly alleviates placement constraints and offers better routability. Based on MCNC benchmark implementation, we showed that the Tree-based architecture has better area efficiency than the common VPR-Style clustered Mesh. Nevertheless, this Tree-based architecture can be 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 particular the Mesh of Tree (a Mesh where clusters local interconnect has a Tree topologie) has a good physical scalability: once the cluster layout is generated we can abut it to generate Mesh layouts with the desired size and form factor. In a future work we are interested to take advantage of both architectures merits by unifying Mesh and Tree interconnects (Mesh of Tree) to get better area efficiency and layout scalability.

6. REFERENCES

[1] E. Ahmed and J. Rose. The effect of LUT and cluster size on deep-submicron FPGA performance and density. *Proceedings of the International Symposium on Field Programmable Gate Arrays*, pages 3–12, 2000.

[2] A.Marquart, V.Betz, and J.Rose. Using cluster-based logic block and timing-driven packing to improve FPGA speed and density. *International symposium on FPGA, Monterey*, pages 37–46, 1999.

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

[4] V. Betz and J. Rose. VPR: A New Packing Placement and Routing Tool for FPGA research. *International Workshop on FPGA*, pages 213–22, 1997.

[5] A. DeHon. Balancing Interconnect and Computation in a Reconfigurable Computing Array (or, why you don't really want 100% LUT utilization). *Proc. FPGA, Montery, CA*, February 1999.

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

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

[9] D. Lewis and al. The stratix logic and routing architecture. *International Symposium on Field Programmable Gate Arrays (FPGA 2003)*, pages 12–20, February 2003.

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

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