Efficient Mesh of Tree Interconnect for FPGA Architecture

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

ABSTRACT

In this paper we present a new Mesh of Tree FPGA architecture, where clusters are surrounded by a Mesh style interconnect and each cluster local interconnect is equivalent to a depopulated Tree-based topology. The particularity of the architecture allows to retain the distinction between Mesh and Tree levels in the mapping phase. This has an important impact on run time saving and tool simplification. Nevertheless an efficient interconnect distribution must be found between both levels, to reach a tradeoff between interconnect reduction and routability. With the proposed Mesh of Tree architecture, we divided the required run time by 3 and reduced the routing interconnect by 24%, compared to the clustered VPR-style Mesh architecture.

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. There exist different ways to connect signals to the Logic Block (LB) inputs. Betz and Rose [1] proposed a fully populated cluster internal interconnect (VPR-style). The fully populated intra-cluster crossbar is simple and ensures a complete local routability. It provides a logical equivalence between clusters in/out pins, with a positive effect on external routability. In addition it allows "distinction" between external and internal nets routing. This has an important effect on routing run time reduction and ensures a strict compliance with the netlist partitioning result. Nevertheless this kind of interconnect does not take advantage of the logical equivalence of LB inputs and has significant area overhead. This penality is increasing especially for architectures with high clusters size. In this paper we show that the intra-cluster full crossbar can be depopulated to achieve significant area reduction without performance degradation. This allowed us to increase the clusters size and to reduce external communication. We propose an improved and less constrained Tree-based interconnect compared to [5] in order to enhance intra-cluster routability. This improvement allows us to experiment a top-down mapping approach: first, we run external (Mesh level) routing and then we run internal (intra-cluster) routing. With such an approach we aim at reducing the external interconnect (Mesh level) while keeping a depopulated intra-cluster interconnect.

In this paper we first present the Tree-based interconnect in a stand-alone mode. All details concerning architeture improvement and configuration flow are described. The Tree architecture is evaluated based on a comparison with VPR-Mesh architecture. Second, we present the architecture unifying Mesh and Tree interconnect and its configuration flow. Finally, an evaluation based on a comparison with VPR-Mesh architecture and a previous studied Mesh of tree [8] architecture is presented.

2. TREE-BASED INTERCONNECT

In a previous work [5] a first hierarchical Multilevel FPGA architecture (MFPGA) was designed and experimentally evaluated. This architecture used a "Butterfly Fat Tree" for the downward network (to connect Switch blocks to LBs inputs) and a limited connectivity Tree for the upward network (to connect LBs outputs to Switch Blocks), following a circular permutation scheme. While providing good flexibility and some interesting features like an almost predictible routing once the placement is defined, this approach revealed some shortcomings hindering highly congested netlists routing:

- The very depopulated upward network, which only allows each LB output to reach any destination by no more different paths than the number of levels in the hierarchy, is detrimental for highly congested netlists.
- The placement of clusters (or LBs for leaf type clusters) inside their owner cluster critically controls routing resources, thus limiting the freedom to re-arrange them and making impossible to construct carry chains in this type of architecture.

2.1 Interconnect improvement

To deal with those weaknesses we propose to add routing flexibility by modifying specifically the upward network. We propose, as shown in figure 1, 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:

- 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 DMSBs reachable per level, with the new upward network, LBs can negotiate with their neighbours (of the

1-4244-1472-5/07/$25.00 © 2007 IEEE

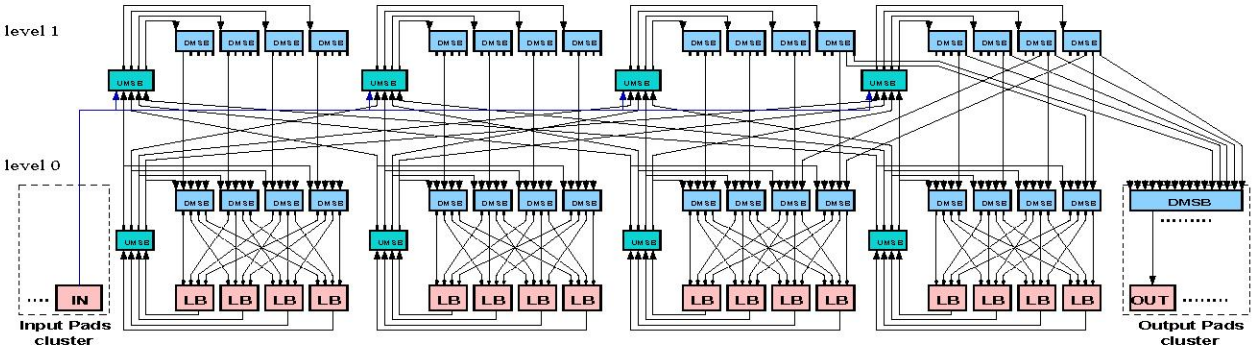

Figure 1: Tree-based interconnect: Upward and Downward Networks

same cluster) 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 1, an LB ouput can reach all 4 DMSBs of its owner cluster at level 0 and all the 16 DMSBs of its owner cluster at level 1.

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 [6] is easily adapted to Tree-based structure ($IO = c.k^{lp}$ 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. By reducing the architecture Rent's parameter the interconnect switches number and routability are reduced. Thus we have to find the best tradeoff between interconnect population and logic blocks occupancy. The occupancy factor is controled by N, the LBs leaves 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 1, 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. Nevertheless there is no logical equivalence between input pads located in different clusters. In fact, pads can reach neighbouring LBs through a greater paths number than the number of paths available to reach LBs located in other clusters. Latitude to permute them remains but with different timing delays.
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. The permutability significantly reduces the effect of pad assignment on Tree-based interconnect routing.

2.4 Configuration Flow

To explore the modified architecture we have to adapt MFPGA configuration flow. Since logic blocks positions inside the owner cluster are equivalent, the detailed placement phase [8] (Arrangement inside clusters) is eliminated. First we apply a top-down multilevel partitioning. In each level hMetis tool [4] is used to create parts with reduced external communication. Routing consists in assigning nets that connect placed instances to routing resources in the interconnect structure. Routing resources are modeled as a directed graph $G(V, E)$. 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" [7],

2.5 Experimental evaluation

To evaluate the different architectures 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. For connection blocks, $F_c = 0.5$ and $F_{c_{out}} = 0.25$ are chosen. We use VPR 4.3 [2] 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, in stand-alone mode, MCNC benchmark circuits. With the previous MFPGA architecture [5] [8], 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. The cost of adding UMSB is compensated by the architecture Rent's parameter reduction (clusters in/out pins number).
We compared also the area requirement between Tree architecture and the clustered VPR-style Mesh architecture to implement these benchmarks. In the case of Mesh we adjust the channel width W and in the case of Tree-based interconnect we adjust levels Rent's parameters and logic occupancy in order to obtain the architecture which best fits each benchmark.
In table 1, we observe that the Tree architecture has a bet-

MCNC	Clustered Mesh Cluster size 4					Tree architecture Clusters Arity 4					Gain		
Circuits	Occup %	Channel Width	SW $\times 10^3$	SRAM $\times 10^3$	Area (λ^2) $\times 10^6$	Occup %	Rent's p	SW $\times 10^3$	SRAM $\times 10^3$	Area (λ^2) $\times 10^6$	SW %	SRAM %	Area (λ^2) %
alu4	86	32	100	74	319	57	0.82	68	55	238	32	26	25
apex4	87	42	359	267	1092	36	0.91	211	151	667	41	56	39
clma	94	51	2541	1879	7672	51	0.86	1484	1058	4781	42	44	38
diffeq	93	29	307	226	954	73	0.81	193	140	619	37	38	35
elliptic	94	41	944	701	2883	43	0.77	552	424	1888	45	40	35
ex5p	92	44	305	224	915	51	0.91	193	143	623	37	36	32
frisc	98	45	952	811	3287	43	0.86	662	478	2164	30	41	34
pdc	93	61	1636	1207	4889	55	0.89	891	620	2785	45	48	43
s38584	96	36	1501	1113	4590	39	0.74	1023	809	3578	32	27	22
spla	96	53	1144	847	3448	45	0.86	721	517	2329	37	39	32
Average	93	43	978	735	3005	50	0.84	600	439	1967	38	39	34

Table 1: Tree vs clustered VPR-style Mesh

ter density and can implement circuits with lower switches number. An average of 38% reduction of the switches number is achieved. We achieve a 32% switches reduction in the case of the "alu4" smallest circuit and 42% in the case of the "clma" largest circuit. This confirms that Tree-based interconnect is very attractive for both small and large circuits and that the best way to improve circuit density, is to balance Logic blocks occupancy and interconnect population [3].

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 34% in the total area compared to Mesh architecture.

3. MESH OF TREE ARCHITECTURE

As shown previously, the Tree-based architecture is more optimized in term of switches requirement than the VPR Mesh architecture. Nevertheless the Butterfly Fat Tree, in stand alone mode, is very penalyzing in term of physical layout generation and wire length especially for large circuits. The idea is to use Tree topology as an intra-cluster interconnect and to use Mesh topology to achieve inter-clusters interconnection. Thus, the architecture we propose has a Mesh of Tree interconnect topology and is built as a matrix of abutted nodes represented in figure 2. Each node has a Tree-based intra-cluster interconnect. The resulting network corresponds to a Mesh of clusters (each one encapsulating the intra-cluster interconnect and the LBs). The mapping of a netlist on this architecture can be executed in two stages:

- The Mesh stage, where clusters are considered as black boxes with i inputs and j outputs. The initial netlist is partitioned into N independent sub-netlists where N corresponds to the number of clusters of the Mesh architecture. Then the inter-clusters netlist (external netlist) is placed and routed using the Mesh interconnect.
- The Tree stage, where each one of the N sub-netlists (internal netlist) is mapped separately in a cluster. In this stage we use the flow described in section 2.4 to place and route sub-netlists

The logical boundary between inter and intra-clusters levels is defined by the Mesh clusters in/out pins which correspond to the Tree in/out pads. In the sequel, the architecture Mesh level and its configuration tools will be described.

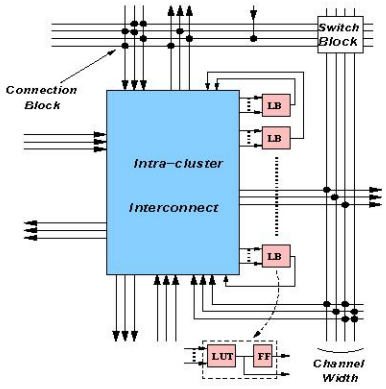

Figure 2: Node of Mesh of Tree architecture

3.1 Mesh routing interconnect

As presented in figure 2, clustered Mesh architecture is composed of logic blocks clusters, switch blocks, connection blocks, and in/out pads. Interconnection between clusters is formed by routes through switch blocks, along horizontal and vertical routing channels. The connection block is the region where the cluster input and output pins connect to the routing channels. Connection block population is defined by $F_{c_{in}}$ 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. In figure 2 $F_{c_{in}} = 0.5$ and $F_{c_{out}} = 0.25$. The switch block is the place where connections are made between the horizontal and vertical routing channels, allowing nets to turn around corners or to extend farther along the channel.

3.2 Top-down configuration flow

The complete configuration top-down flow is desribed on figure 3. First, the netlist instances are partitioned between Mesh clusters. Second, the obtained external netlist (inter-clusters) is placed and routed on the Mesh using VPR [2]. After the inter-clusters netlist routing, clusters in/out pins are assigned to specific signals. Those signals are considered as in/out pads in the generated intra-cluster netlists. Thus, as presented in figure 3, external netlist routing as-

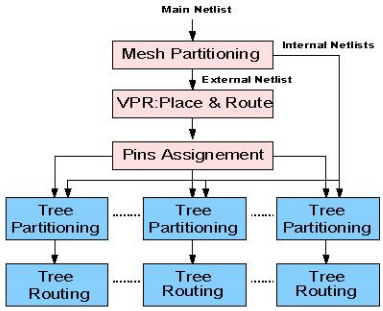

Figure 3: Top-down Mesh of Tree configuration flow

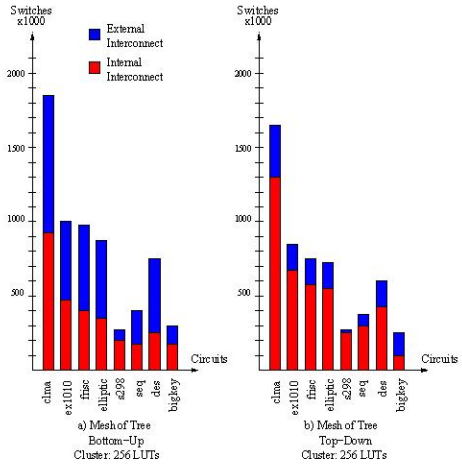

Figure 4: Internal and external interconnect distribution in Mesh of Tree architecture

signs the specific position of each internal netlist pad (pin assignment). Those positions are strictly respected when we place and route an internal netlist on the Tree-based architecture.

3.3 Experimental results

3.3.1 Mesh of Tree: top-down vs bottom-up

Here we compare the Mesh of Tree architecture based on the improved Tree architecture, where the mapping uses the top-down approach, to the previous bottom-up mapping approach presented in [8] (using the old Tree architecture [5]). In both cases, we use clusters containing 256 LBs and we compare the required architecture characteristics to implement the same circuits.

With the top-down approach we reduce the required Mesh channel width. The external interconnect switches reduction can be seen in figure 4. The external switches are reduced by 58%. Nevertheless, the intra-clusters interconnect (Tree level) were increased to get more flexibility to satisfy the pins assignment constraints. Consequently, the internal switches number is increased by 43%. Using the top-down approach, we have achieved on average a total switches reduction equal to 14%.

3.3.2 New Mesh of Tree vs VPR-style Mesh

We compared switches requirement in the case of clustered VPR-style Mesh and the proposed Mesh of Tree architecture (top-down approach). We notice that with the Mesh of Tree architecture we obtain a better density and about a 24% reduction on the number of switches. Thanks to the Mesh large clusters size (256), the required mapping run time is divided by 3.

4. CONCLUSION

The improved Tree-based architecture alleviates significantly placement constraints and allows in/out pins permutability. We took advantage of this fact and proposed an improved architecture combining Mesh and Tree merits. The architecture configuration approach retains distinction between both levels. As we showed, routing independently inter-clusters netlists (Mesh level) and intra-clusters netlists (Tree level) is no longer penalizing and allows a top down approach. Indeed, both run time and switches are reduced without routing degradation.

We notice from table 1 and figure 4 that the Tree-based architecture is the most optimized architecture in term of switches requirement (better than the Mesh of Tree). Nevertheless this Tree-based architecture, in stand-alone mode, is very penalizing in term 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 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 form factor. In addition in this case we consider small Tree-based interconnect (256 BLEs) with reduced wires lengths.

5. REFERENCES

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

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

[3] 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.

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

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

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

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

[8] Z.Marrakchi, H.Mrabet, C.Masson, and H.Mehrez. Mesh of Tree: Unifying Mesh and MFPGA for better Device Performances. *International Symposium on Network-on-Chips, Princeton, New Jersey*, May 2007.