Performances improvement of FPGA using novel multilevel hierarchical interconnection structure

Hayder Mrabet, Zied Marrakchi,Pierre Souillot and Habib Mehrez
LIP6, Université Pierre et Marie Curie
4, Place Jussieu, 75252 Paris, France

ABSTRACT

This paper presents a new Multilevel hierarchical FPGA (MFPGA) 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 the Rent's Rule show that wiring and switch requirements in the MFPGA grow slower than in traditional topologies. New tools are developed to place and route several benchmark circuits on this architecture. Experimental results based on the MCNC benchmarks show that MFPGA can implement circuits with an average gain of 40% in total area compared with mesh architecture.

1. INTRODUCTION

Earlier Field Programmable Gate Arrays(FPGAs) provided a sea of Look-Up Tables(LUTs) and registers which are linked together using programmable interconnections. Several topologies of programmable interconnect like the Manhattan mesh based FPGAs [3], the hierarchical FPGAs [1, 7, 13], and the hierarchical mesh architecture [5, 12] have been proposed. These investigations present the networks characteristics and how they scale.

Driven by Moore's law semiconductor scaling, larger and larger FPGAs emerge. Current architectures will not extend directly to this scale (the one-million gate and more) because routing requirement and delays grow linearly. In addition, placement and routing computational times are ever increasing nowadays. Excessive FPGA placement and routing runtimes are now often measured in hours.

Design of large devices imposes radical efficient change in architecture to improve speed, density and software mapping time. Relying on industry experience with standard ASICs, we believe that partitioning and hierarchy becomes an unavoidable for hardware and software developments. As an alternative we propose a new multilevel hierarchical FPGA (MFPGA) architecture where logic blocks and routing resources are sparsely partitioned into a multilevel clustered structure.

ICCAD 06, November 5-9, 2006, San Jose, CA

In this paper we examine the state of the art and then we detail our proposed architecture and preliminary results using specific tools. Next section describes the state of the art and compares traditional symmetric manhattan mesh with hierarchical architectures. Section 3 describes our new MFPGA architecture and evaluates its wiring and switch requirement growth. In section 4 we propose suitable techniques to place and route the MFPGA with experimental results using the specific tools.

2. PREVIOUS WORKS

Mesh is the most studied and used industrial topology. This approach has been the subject of considerable published work by Rose et al. [3]. Mesh is a regular structure in Island style. It consists of an array of logic blocks with I pins in each side, which can be linked together by uniform horizontal and vertical programmable routing channels. Each routing channel contains W wire tracks. Each pin of the logic block may be connected to a Fc fraction of the wire tracks in the adjacent routing channel through a connection box. A switch box is located at vertical and horizontal channels crossings. Each track entering the switch box may connect Fs tracks (here we treat the case of depopulated switch box such as 'subset' or 'universal' [3]).

There is little published research on Hierarchical FPGA (HFPGA) architecture [13, 7, 1]. Part of the reason is that industrial leaders use the manhattan mesh architecture. In a hierarchical FPGA , logic blocks and routing resources are organized into levels. At each level there are blocks and routing resources belonging to that level. A $level_i$ cluster has W_i IO (Input/Output) wire tracks and contains a set of k $level_{i-1}$ clusters connected with $level_i$ switch box. A common way to compare both architectures is the wiring requirement using Rent's Rule [8].

$$IO = cN^p$$

This empirical relationship links the number of IO with the N gates of a design. For our case of FPGA, N corresponds to the number of logic blocks linked together. c is a constant factor that corresponds to the IO size of the logic block, and p defines the growth rate of the rent's rule.

Thanks to this rule and the bisection method presented in [7], DeHon [5] links the channel width parameters W of a mesh arranged as $\sqrt{N} * \sqrt{N}$ logic blocks to the Rent parameters. Therefore the Rent's Rule provides a lower bound of W to support a design characterized by Rent parameters (c, p)

$$W \geq \left(\frac{c}{2^p}\right) N^{p-0.5} \quad (1)$$

Since we know the topology of the mesh architecture and its specific parameters, we can deduce the total number of switches per logic block in a mesh

$$N_{switch} = 4W(F_c I + F_s) = O(W) = O(N^{p-0.5}) \quad (2)$$

The number of wire per logic block :

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

Rent's Rule is easily adapted to hierarchical structure. A level i cluster contains k^i logic blocks and therefore has ck^{ip} IO. Hence:

$$W_i = ck^{ip} \quad (4)$$

$$N_{wire} = \begin{cases} O(log_k(N)) & \text{if } p = 1 \\ O(1) & \text{if } p < 1 \end{cases} \quad (5)$$

As the k-HFPGA uses full cross bar switch boxes, the switching requirement is evaluated as follow:

$$N_{switch} = \begin{cases} O(N^{2p-1}) & \text{if } p > 0.5 \\ O(log_k(N)) & \text{if } p = 0.5 \\ O(1) & \text{if } p < 0.5 \end{cases} \quad (6)$$

Equations (6) and (5) compared respectively to equations (2) and (3) show that k-HFPGA has more efficient switching and wiring area that grows as $O(1)$ if $p < 0.5$. When $p > 0.5$ the mesh architecture becomes more interesting. Mesh switching and wiring area grows as $O(N^{p-0.5})$ while the k-HFPGA switching resources diverge and grows as $O(N^{2p-1})$. Unfortunately for the k-HFPGA, a large number of authors observe that typical designs have $0.5 \leq p \leq 0.75$. Thus the mesh architecture is still more efficient.
We note that the above k-HFPGA is penalized by the use of a full cross bar switch box that guarantee arbitrary, full connectivity, thus overestimating the required number of switches. In spite of its low logic density, the k-HFPGA retains the advantage of lower routing delays. There is a strong reduction concerning the average number of crossed switches to connect two logic blocks. The worst case is proportional to the number of levels in the tree $O(\log_k(N))$.
Both Agarwal [1] and Lai [7] described hierarchical FPGA interconnect architectures with a sparse depopulated switch box. Both topologies tend to depopulate the switch patterns while maintaining 100% routability or at least the same routability as the depopulated linear mesh architecture. Tsu [13] describes the HSRA architecture based on the butterfly architecture that needs fewer switches than in the other cases. In general, such binary tree is a limiting interconnect structure that leads to severe routing inefficiencies.

3. PROPOSED ARCHITECTURE

We propose a new reprogrammable Multilevel hierarchical FPGA (MFPGA) architecture, designed to have very low cost compared to traditional FPGA. The main motivation for the new architecture is to achieve the best area efficiency by balancing interconnect and logic block utilization. In [4] DeHon answers the question *"Is an FPGA with higher LUT usage more area efficient than one with lower LUT utilization ?"* and concludes that we must under-utilize first resource in order to fully use the other. Since the routing resources consume most of the area (often 80-90%), we focus on interconnect check. This suggests that mapped circuits will be routed sparsely on the interconnect, leaving many logic blocks unused.
All networks presented in the previous section use bidirectional switches and wire tracks. This introduces considerable complication in both hardware network design and effort deployed by routing tools. Innovation in this new architecture consists in the use of unidirectional switches. We distinguish two networks:

- The downward network inspired from SPIN [6], based on the Butterfly Fat-Tree(BFT) style interconnect [9] with linear populated and unidirectional switch boxes. The leaves of the hierarchy are logic blocks.
- The upward network that connects the logic blocks outputs and the Input Pads to the different levels of the Downward Network .

Each logic block contains one 4-input Look-Up-Table (4-LUT) followed by a bypass Flip-Flop. This 4-LUTs is shown to be the most efficient K-LUT for SRAM based FPGAs by J.Rose et al in [2].
We use the Rent's parameters to specify the bandwidth growth for each network.
We assume that p is the same for both networks. Likewise, as mentioned previously we use logic blocks whith 4 inputs and 1 output. According to Rent's Rule, if we are limited to one logic block, $N_{in} = c_{in}(1)^p$ inputs and $N_{out} = c_{out}(1)^p$ outputs. Thus we obtain both values of c:
- $c = 4$ for the downward network.
- $c = 1$ for the upward network.

3.1 The Downward Network

A configuration of a n level MFPGA can be described using the expression N_0 x N_1 x N_2 x....x N_{n-1}. Let's take the example of a 4x4x4 MFPGA architecture with 64 logic blocks.
This is a 3 levels Downward Network (or BFT tree) with the following parameters: $p = 1$ and $k = 4$ (the arity of the BFT).
The lowest cluster shown in figure 1 is composed essentially of 4 Logic Blocks (LBs) and one switch box composed of 4 Mini Switch Boxes (MSBs). Each MSB is in charge of the interconnection between the upper level and one input of each logic block.
Thus the MSB has 4 outputs and, as we deal with unidirectional wires, they are composed of 4 multiplexers, one for

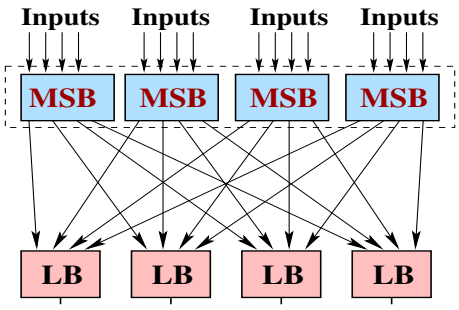

Figure 1: Downward Network in the Level 0 Cluster

each output.
This cluster has 4 Logic Blocks so it must have $c_{in} * (k)^p = 4 * 4^1 = 16$ inputs distributed on the four MSBs.
In the same manner we create the $level_1$ cluster. We connect 4 $level_0$ clusters to one Switch Box. As the $level_0$ cluster has 16 inputs, we divide the $level_1$ switch box in 16 MSB. Each MSB has 4 outputs, each output will be connected to one $level_0$ cluster input. This super cluster contains 16 Logic Blocks so it has $c_{in}k^{2p} = 64$ inputs shared out among the 16 MSBs.
The last level is created likewise : We connect 4 $level_1$ clusters to one switch block. As each $level_1$ cluster has 64 inputs, we divide the Switch Block in 64 parts and connected each to one input of each $level_1$ cluster.

3.2 The Upward Network

The Downward Network gives one path from each wire-source in the top to each leaf (logic block) in the lowest level. Now what can be the source ? Of course it will be a logic block output or an input pad.
Let us begin with the logic block output. How to connect the output of a logic block to the input of another one ? We just have to bring the logic block output signal to a specific upper level (the lowest authorized level is the lowest level common to the two logic blocks), then the signal can go down to the targeted logic block through the BFT. We name these signals $feedbacks$. If we chose $p = 1$, the number of outputs in a $level_0$ cluster corresponds to the number of logic block in this cluster. We therefore connect each output to one MSB at each level. Therfore each logic block output will be connected to n different MSBs, one in each level (n is the number of level).
The way we distribute the feedbacks between levels has an important impact on the structure routability. Connecting a feedback to n MSBs with different indexes increases the paths from one source to one destination. We did it as simply as possible and we define the solution presented in figure 2-(a). This feedbacks distribution gives different possibilities to reach a destination. Since we can permute the logic block inputs, the goal is to reach the destination logic block on any input pin. Figure 2-(b) shows how the logic block 'A' output can reach the logic block 'B'. Each level offers one path from 'A' to 'B', the three paths reach three different 'B' input pins. All input pins of the logic block are similar since we use a LUT. It is easy to adapt the LUT mask to any permutation of the four LUT inputs.

3.3 Connections with the Outside

Both Input pads (In) and Output pads(Out) are clustered with the logic block at $level_0$. In the Example of 4x4x4 we use 64 Input and 64 Output. Thus each $level_0$ cluster contains 4 Inputs and 4 Outputs.
The number of IO pads per cluster can be varied to obtain the best design fit. We use a local interconnect between the logic block outputs and the output pads. Furthermore we can connect the Output pads to the downward network. To reduce complexity, we eliminate the last case at this stage.
Concerning input pads, we connect them to the upward network just as the logic blocks outputs. Input pads can reach any local logic block directly through the local switch box, and can reach other logic blocks through common levels.
Figure 3 shows the topology used in the 4x4x4 MFPGA with

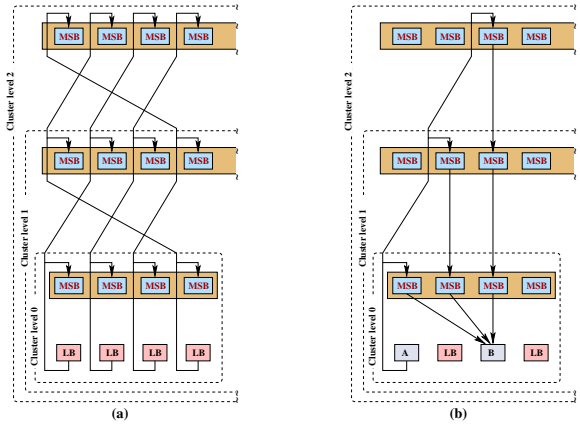

Figure 2: The Impact of the Upward Network

4 Input pads and 4 output pads in each $level_0$ cluster. Because in the 4x4x4 MFPGA we don't introduce constraints on the outputs placement, and the number of output pads is equal to the logic block number, we connect each logic block output to one output pad.
The overall MFPGA with $p = 1$ and $k = 4$ is given in Fig.3.

3.4 Rent's Rule based MFPGA model

As we parametrized our architecture we can now evaluate the number of wires and switches per logic block. Consider a k-arity MFPGA as depicted above with N Logic Blocks and whose wire growth follows the Rent's Rule. Each Logic Block has c_{in} inputs and c_{out} outputs.
A $level_i$ cluster contains $N_{in}(i-1)$ MSB with k outputs and $\frac{N_{in}(i)+kN_{out}(i-1)}{N_{in}(i-1)}$ inputs. If we assume that the MSB are full cross bar device and have an upper bound on the switches number, we have $k(N_{in}(i) + kN_{out}(i-1))$ switches in the switch box of a $level_i$ cluster. As we have $\frac{N}{k^i}$ $level_i$ clusters in the overall structure we have a total number of switches of :

$$\sum_{i=0}^{\log_k(N)-1} kN\frac{N_{in}(i) + kN_{out}(i-1)}{k^i}$$

$N_{out}(-1) = c_{out}$ is the number of outputs of a Logic Block. Following (4), we have $N_{in}(i) = c_{in}k^{(i+1)p}$ and $N_{out}(i) = c_{out}k^{(i+1)p}$.
The Number of switches per Logic Block is :

$$N_{switch} = k \sum_{i=0}^{\log_k(N)-1} (k^p c_{in} + kc_{out})k^{i(p-1)}$$

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

$$N_{switch} = \begin{cases} O(1) & \text{if } p < 1 \\ O(\log_k(N)) & \text{if } p = 1 \end{cases} \quad (7)$$

Here we don't take into account the switches (per logic block) needed by the local interconnect for the output pads. Their number is negligible since it is in $O(io\_ratio)$. io_ratio is the number of IO pads per $level_0$ cluster.
For each $level_i$ cluster, $(kN_{out}(i-1) + kN_{in}(i-1))$ wires

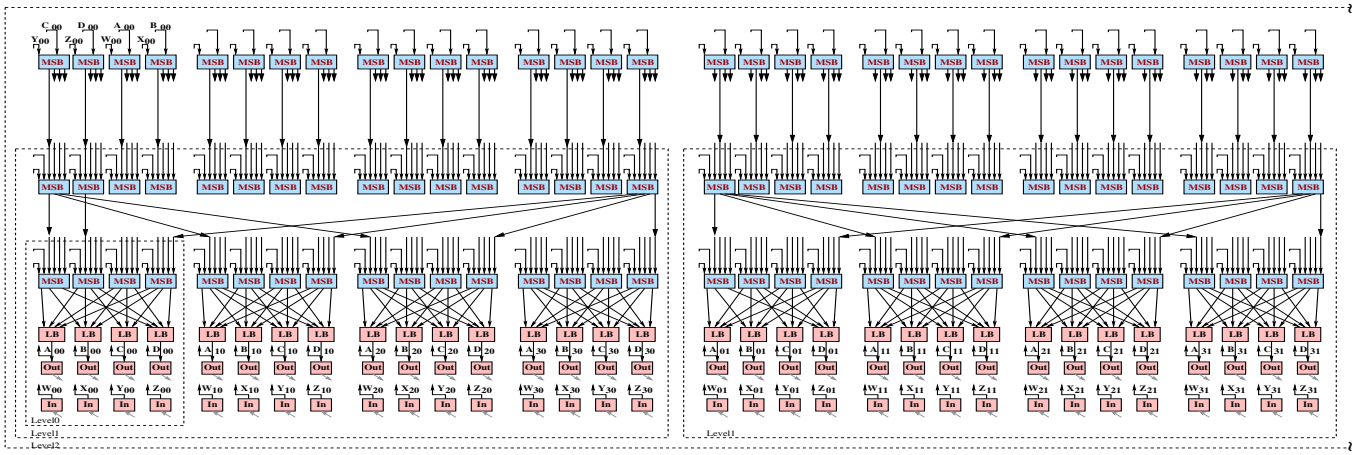

Figure 3: Three level MFPGA structure with $k = 4$ and $p = 1$

are connected to the MSB. Thus the total number of wires per block is:

$$N_{wire} = \sum_{i=0}^{\log_k(N)-1} (c_{in} + c_{out})k^{1+i(p-1)}$$

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

$$N_{wire} = \begin{cases} O(1) & \text{if } p < 1 \\ O(\log_k(N)) & \text{if } p = 1 \end{cases} \quad (8)$$

Equations (7) and (8) show that both switch and wiring requirement grow more slowly than in traditional architectures described in section. 2. These results are encouraging for the construction of very broad MFPGA structures especially when p is less than one. But this does not mean that our MFPGA topology is more efficient than other architectures since they do not have the same routability. The best way to check this way is to launch experimental work and compare the area results using MFPGA and the Manahttan FPGA.

4. EXPERIMENTAL RESULTS

As explained previously our routing resources are limited and we have a few different ways to connect a source to a destination. Consequently the placement of the cells has an important impact on the netlists routability. In fact most part of the effort will be devoted to the placement phase, which is done in two steps. First we apply a global placement. The aim of this phase is to balance the nets to route between clusters. It consists of a multilevel clustering and a multilevel refinement phases. Second in each level we run a detailed placement to select slots that will be occupied inside clusters. We adopted a particular iterative rip-up algorithm based on congestion negotiation called PathFinder [11] for the routing phase. Methods and algorithms used for placement and routing phases are described in [10].

To validate and study the performance of our tools, we placed and routed a set of MCNC benchmark circuits. As shown in table 1 results are very promising since we were able to route circuits that occupies until 77% of the logic area.

We use the same benchmark circuits to compare the switch and area requirements between our MFPGA architecture and traditional mesh topology. The mesh is similar to the vpr422_challenge_arch architecture with uniform routing with single-length segments and a subset switch box. Each Logic Block contains only one 4-LUT. One input appears on each side, and the output appears on the top and the right side. Both inputs and outputs are fully populated ($F_c = 1$), and IO pads are fully populated too.

We use the channel minimizing VPR 4.3 router to route the mesh, and we vary the IO_ratio to achieve the optimal array size.

VPR choose the optimal size as well as the optimal channel width needed to place and route each benchmark. For the MFPGA we choose a structure large enough to support the benchmark circuit. MFPGA structures can be varied by changing the level number, the arity of each level. Rent's parameters p is fixed at 1.

In both cases the number of switches needed by each benchmark corresponds to the total number of switches used by the overall optimal target architecture.

We compare the areas of both architectures using successively a simple cost model based on routing switch counts, and a more refined model that more accurately estimates 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 MFPGA, composed primarily of SRAMs and Multiplexers. We use the same cells library for both architectures.

Table 1 summarizes the basic results for the Mesh and for the MFPGA.

Given a benchmark of some fixed size, and performing an experiment of MFPGA with specified parameters, we get results as summarized in the right part of table 1. Column 9 shows the occupation average of each circuit in the target MFPGA. There is a low occupation average in the majority of the benchmarks. This is due to the depopulation of the interconnect. As mentioned previously we under-utilize the logic resources in this type of structure. In addition, size of smallest MFPGA that can contain the circuit under investigation is penalized due to the coarse granularity of this architecture. In spite of these constraints we achieve a

Benchmark		Mesh					MFPGA $p = 1$				
Name	LUTs	$\sqrt{N}$	W	IO ratio	Switches number	Area (λ^2) x10^3	Arch	Occup-ation%	Routing success%	Switches number	Area (λ^2) x10^3
b1	4	2	3	2	300	1284	4	100	100	32	288
cm138a	9	3	4	2	824	3344	4x4	56	100	512	2032
cm42a	10	4	3	1	948	4344	4x4	63	100	512	2032
pcle	29	6	5	2	3700	15316	4x2x2x4	46	100	3584	11968
decod	32	6	4	1	2768	11822	4x4x4	50	100	3584	11648
cc	33	6	5	2	3700	15316	4x4x4	52	100	3584	11648
count	37	7	5	2	4950	20577	4x4x2x2	58	100	4096	12608
my_adder	49	7	4	2	3960	16680	4x4x4	77	100	3584	11648
b9	61	8	5	4	7020	28656	4x4x4	96	98	3584	11648
i4	110	11	7	5	18298	71289	4x4x4x4	42	100	20480	46080
c2670	363	20	8	5	63968	249172	4x4x4x4x4	35	100	106496	299008
i9	471	22	8	2	72480	286356	4x4x4x4x4	46	100	106496	299008
alu4	584	25	8	1	91568	363547	4x4x4x4x4	57	100	106496	299008
tseng	1085	33	9	1	178758	709785	4x4x4x4x4x2	53	100	253952	679936

Table 1: Benchmark statistics

gain in area efficiency compared to the mesh architecture. Columns entitled "Switches number" and "Area" in table 1 show the difference in number of switches and total area in Mesh and the MFPGA structures respectively.

It is clear from this comparison that the new architecture will be more efficient in terms of area if we can increase the Logic utilization. We see that for a low occupation average of the logic resources(LBs), the whole MFPGA device is smaller than its equivalent mesh since it requires less interconnect. In other hand, as the occupation grows, the routing complexity grows and needs more resources.

5. CONCLUSION

This paper describes a new hierarchical multilevel MFPGA architecture. We show that good balancing of LUT utilization and interconnect utilization implies lower area than the traditional Mesh.
The preliminary results fully indicate that we meet the capacity and performance targets.
The new topology based on two hierarchical unidirectional networks seems more robust and can achieve better speed than symmetrical FPGA architectures. The average path length between two LBs grows as $\log_k(N)$ for an MFPGA whereas it is $O(N)$ for a Mesh.
The downward network is a predictable interconnect, due to unicity of the downward path leading to destination. This particularity yields a very interesting benefit for the routing phase.
Our analysis does not cover all possible cases, specially for larger arity $k = 8$ or $k = 16$ and Rent's parameter $p < 1$. This is still a fertile research area for this new MFPGA architecture.

6. REFERENCES

[1] A. Aggarwal and D. Lewis. Routing Architectures for Hierarchical Field Programmable Gate Arrays. *Proc. 1994 IEEE. Int. Conf. Computer Design*, October 1994.

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

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

[4] 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.

[5] A. DeHon. Unifing Mesh and Tree-Based Programmable Interconnect. *IEEE Transactions on VLSI Systems*, IEEE Transactions on VLSI Systems(12):10, October 2004.

[6] P. Guerrier and A. Greiner. A generic architecture for onchip packet-switched interconnections. *Proceedings of the Design Automation and Test in Europe Conference 2000 (DATE 2000), Paris, France*, page 250 256, Mars 2000.

[7] Y. Lai and P. Wang. Hierarchical Interconnection Structures for Field Programmable Gate Arrays. *IEEE Transactions on VLSI Systems*, 5(2), June 1997.

[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] C. Leiserson. Fat-trees: Universal networks for hardware efficient supercomputing. *IEEE Transactions on Computers*, C-34(10):892–901, October 1985.

[10] Z. Marrakchi, H. Mrabet, and H. Mehrez. A new Multilevel Hierarchical MFPGA and its suitable configuration tools. *Proc. ISVLSI-2006, Karlsrhue, Germany*, March 2006.

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

[12] R. Rubin and A. DeHon. Design of FPGA interconnect for multilevel metallization. *IEEE Trans. VLSI Syst*, 12:1038–1050, Oct 2004.

[13] W. Tsu. et al. HSRA: High Speed, Hierarchical Synchronous Reconfigurable Array. *Proceedings of the International Symposium on Field Programmable Gate Arrays*, pages 125–134, February 1999.