International Journal of Electronics
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713599654

Performances comparison between multilevel hierarchical and mesh FPGA interconnects

Z. Marrakchi [a]; H. Mrabet [a]; C. Masson [a]; H. Mehrez [a]
[a] LIP6, Université Pierre et Marie Curie, Paris, 75005, FR

Online Publication Date: 01 January 2008

To cite this Article: Marrakchi, Z., Mrabet, H., Masson, C. and Mehrez, H. (2008) 'Performances comparison between multilevel hierarchical and mesh FPGA interconnects', International Journal of Electronics, 95:3, 275 — 289

To link to this article: DOI: 10.1080/00207210701828069
URL: http://dx.doi.org/10.1080/00207210701828069

International Journal of Electronics,
Vol. 95, No. 3, March 2008, 275–289

Performances comparison between multilevel hierarchical and mesh FPGA interconnects

Z. MARRAKCHI*, H. MRABET, C. MASSON and H. MEHREZ

LIP6, Université Pierre et Marie Curie, 4 place Jussieu, Paris, 75005, FR

(*Received 11 February 2007; in final form 26 November 2007*)

This paper evaluates a new multilevel hierarchical FPGA (MFPGA). The specific architecture includes two unidirectional programmable networks: a downward network based on the Butterfly-Fat-Tree topology; and a special upward network. New tools are developed to place and route several benchmark circuits on this architecture. Comparison with the traditional symmetric Manhattan mesh architecture shows that MFPGA can implement circuits with a smaller area and better speed.

Keywords: FPGA; Interconnect; Hierarchical; Partitioning; Routing

1. Introduction

Earlier field programmable gate arrays (FPGAs) provided a sea of look-up tables (LUTs) and registers which are linked using programmable interconnections. Several topologies of programmable interconnect like the Manhattan mesh based FPGAs (Betz *et al*. 1999) and the hierarchical FPGAs (Aggarwal *et al*. 1994, Laiand Wang, 1997, Tsu *et al*. 1999) have been proposed. These investigations present the networks characteristics and how they scale.

Driven by Moore's law on semiconductor scaling, ever 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 constantly 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 become 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.

This paper details the proposed architecture and preliminary results using specific tools. Section 2 describes the new MFPGA interconnect architecture. Next suitable techniques are proposed to place and route applications on MFPGA. In the

*Corresponding author. Email: Zied.Marrakchi@lip6.fr

International Journal of Electronics
ISSN 0020–7217 print/ISSN 1362–3060 online
http://www.tandf.co.uk/journals
DOI: 10.1080/00207210701828069

experimental results section, MFPGA performances are compared to the common mesh architecture using MCNC benchmark circuits.

2. Architecture overview

A standard hierarchical FPGA is denoted k-HFPGA, in which a cluster has k subclusters. The structure can be presented by a tree. Figure 1 is an example of a 4-HFPGA where a cluster contains 4 subclusters. A vertex in this tree is used to represent a logic or a switch block. An edge between two vertices is used to represent a routing channel which consists of a set of tracks. The logic blocks are at the bottom of the tree while the switch boxes are those vertices above the logic blocks. A modified multilevel hierarchical architecture denoted MFPGA, which can be more interesting in terms of area and performances, is proposed, which can be more interesting in terms of area and performances. The architecture has the following particularities.

- The lowest level of the hierarchy contains the Logic blocks and the IO pads. Each logic element contains one 4 inputs Look-Up Table (4-LUT) followed by a bypass Flip-Flop.
- The routing architecture contains only unidirectional wires and the switch boxes are depopulated.
- In each level, the ratio between parent tracks and child tracks is equal to k (k is the number of slaves in the cluster).

As we use unidirectional switches, we can distinguish two connecting networks as shown in figure 1.

- A downward connecting network whose topology is equivalent to the butterfly fat tree Leiserson (1985). In this tree, the edges come from the upper levels and reach the inputs of the logic blocks. The topology of this tree is equivalent to the one used in SPIN network Guerrier and Greiner (2000).
- An upward connecting network whose edges come from the leaves (outputs of logic blocks and input pads) to the switch boxes of each level.

2.1 *Downward network*

Consider the case of a 2 levels tree with an arity equal to 4. In each level, a cluster contains 4 slaves and a switch box. To depopulate the switch box, we divide it into

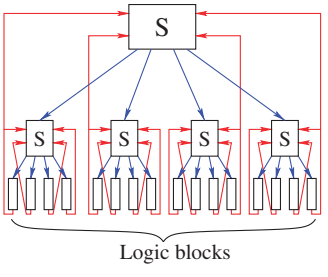

Figure 1. Connection networks.

4 mini switch boxes (MSB). In level 0, each MSB is in charge of connecting the upper level tracks and one input of each logic block as depicted in figure 2. Thus, each MSB has 4 outputs which are equal to the number of logic blocks (slaves). Level 1 is constructed in the same manner, the switch box of each cluster of level 1 is connected to 4 clusters belonging to level 0. As each cluster in level 0 has 16 inputs, the switch boxes are divided into 16 MSB and each one is connected to one cluster slave input.

Figure 3 shows the distribution of the interconnect in level 1. The previous described butterfly fat tree has the following properties.

- From a track located in the top of a switch box one can reach any slave, but in only one pin.
- From a track of a switch box there is only one path to reach a particular slave. Due to the regularity of the architecture, this path is easily predicted.
- In each level, the interconnect resources are balanced between clusters.

2.2 *Upward network*

It is proposed to connect logic blocks output signals to specific switch boxes of upper levels. Thus, for each logic block output (and input pad), a list of feedbacks is defined, each one enabling the output to reach a switch box on a particular level.

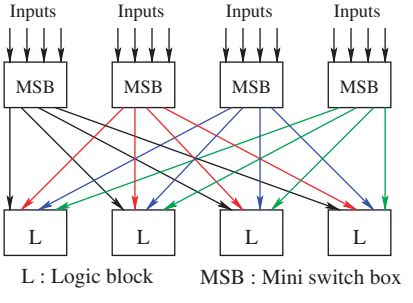

Figure 2. Top-down connecting tree in level 0.

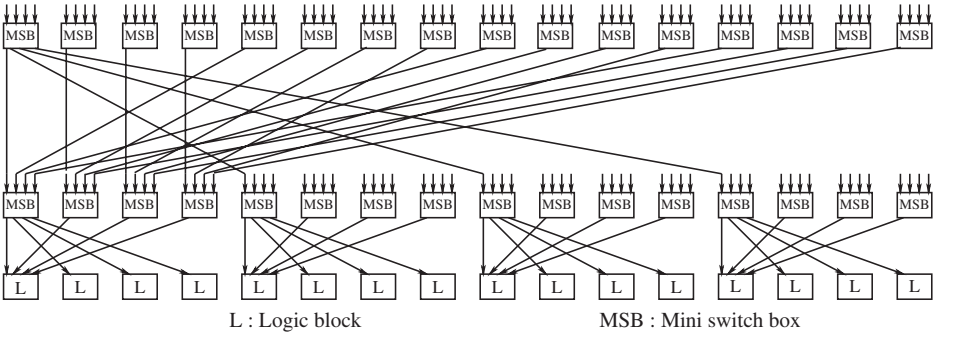

Figure 3. Top-down connecting tree in level 1.

The way in which feedbacks are distributed on each level has an important impact on the number of different paths to reach a destination logic block from a source. As shown in figure 4, two feedbacks can connect a source to a destination using two different paths.

3. Placement

The MFPGA placement problem can be stated as assigning to each netlist cell a logic block (leaf) in the MFPGA architecture. The way cells are distributed also has an important impact on routability. In fact, once cells are placed, the router tries to find a path to connect a source LB (cluster leaf) to its destinations LBs (cluster leaf) using architecture resources. Thanks to the interconnect predictibility provided by the MFPGA architecture, some conditions to limit later conflicts in the routing phase can be introduced in the placement phase.

3.1 *Conflict condition*

To present the effect of placement on routability, refer to figure 5. This figure presents two different placement cases of the same netlist. In this netlist, *cell*1 and *cell*2 drive *cell*0. In the first placement case, a conflict (dashed line) is obtained to route the netlist using the lowest levels. Changing the position of *cell*2 resolves the routing conflict problem (second placement case).

To consider routing constraints in the placement phase, a sufficient condition is established to avoid conflicts occuring.

Lemma 1: If $pos(cell) - pos(cell') \neq (l'_{up} - l_{up})\ mod\ 4$ there is no conflict between *cell* and *cell*′ to reach a common destination using respectively l_{up} and l'_{up} levels.

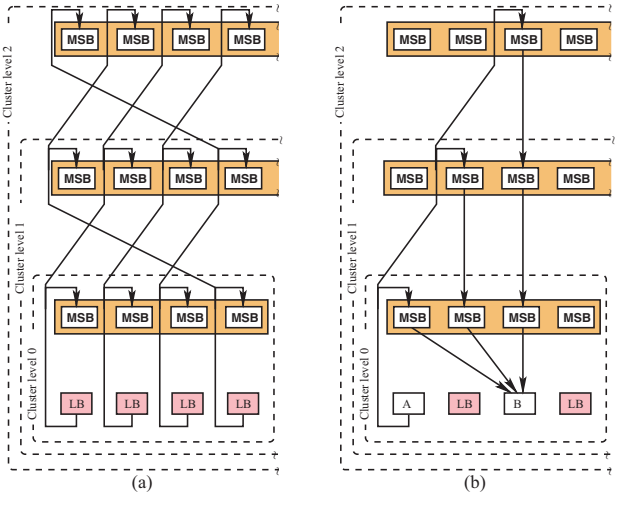

Figure 4. The upward connecting network.

This Lemma is confirmed by placement examples presented in figure 5. In the first example, (unfeasible routing), $pos(cell1) - pos(cell2) = 3$ and $l2_{up} - l1_{up} = 3$ (in base 4). This equality explains why a conflict in this placement case is obtained. In the second example, (feasible routing), $pos(cell1) - pos(cell2) = 2$ and $l2_{up} - l1_{up} = 3$. As Lemma 1 is verified, netlist routing with no conflict is achieved.

3.2 *Partitioning*

The way in which logic blocks are distributed between MFPGA clusters has an important impact on routing congestion reduction. Based on the upward interconnect specificity notice that the number of different paths to connect a source to a destination depends on their enclosing clusters. If they are packed in the same cluster, the source can use more levels to reach its destination (more different paths). From this remark, consider that the netlist cut reduction is an important factor for routability improvement.

A second partitioning objective is deduced from routability condition presented in Lemma 1. Consider a netlist where *cell* and *cell′* are driving a common destination. If both sources are packed in the same cluster, one obtains on the one hand, $\ell_{up} - \ell'_{up} = 0$ (ℓ_{up} is the lowest used level by *cell* to reach the destination); on the other hand $pos(cell) - pos(cell') \neq 0$. In this case, referring to Lemma 1, no resource conflict occurs.

To include this objective in the clustering technique, it is proposed to construct a cells constraints graph (CCG). The CCG denoted as $G_n = (V, E_n)$ consists of a set of vertices and weighted edges derived from the netlist. An edge is established between two vertices when they drive the same destination cell, which are called adjacent. Each edge contains a weight equal to the number of common destinations between two adjacent vertices. Using only this graph in the partitioning weakens the obtained clusters netlist results in terms of external communication. To take this in consideration, it is proposed to generate from the initial netlist hypergraph and the CCG a new constrained cells hypergraph CCH, as presented in figure 6. In this hypergraph, vertices are cells (as in the netlist) and it contains all hyperedges of the netlist and all edges of the CCG. This constrained hypergraph is partitioned by hMetis Karypis

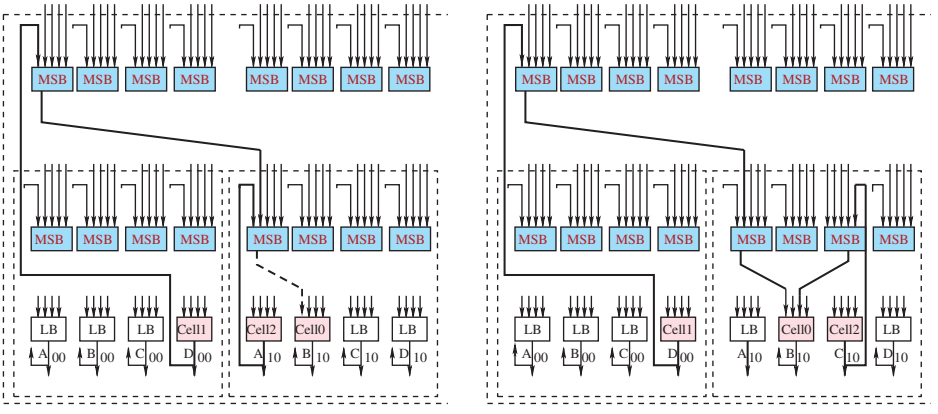

Figure 5. Placement example.

and Kumar (1999) partitioning tool and objectives priorities are defined by hyper-edges weights.

It is proposed to use a top-down patitioning approach (global connectivity informations). First construct clusters of the top level and then each cluster is partitioned into sub-clusters. This is done until the bottom of the hierarchy is reached. To run partitioning hMetis is used, since it generates a good solution in a short time due to its multi-phase refinement approach.

3.3 *Detailed placement*

If during detailed placement Lemma 1 condition is taken into account, significant gain can be obtained in terms of routability and congestion reduction. For this purpose, one introduces the advanced cells constraints graph (ACCG), with which is associated a given a multilevel clustered netlist (previous section) and a placement problem. Input and output Pads are also concerned by the detailed placement. To simplify method description, we consider only logic blocks. Input pads are treated exactly like logic blocks. Output pads are clustered with their logic block drivers.

3.3.1 Advanced cell constraints graph. An ACCG is a CCG that contains extra cells partitioning informations required to check conditions of Lemma 1. An ACCG denoted as $G_n = (V, E_n)$ consists of a set of vertices and directed edges derived from the netlist and the way its cells are partitioned between clusters in each level, where each vertex corresponds to a cell of the netlist. A pair of opposite directed edges is established between two vertices when they drive the same destination cluster (located at any level), which are then called adjacent. To be able to verify conditions proposed in Lemma 1, it is necessary to add further information to the constraints graph. This information is stored in each directed edge connecting two adjacent vertices and consists of the forbidden shift between adjacent vertices positions.

$$shift = (\ell_{up} - \ell'_{up}) \ mod \ 4$$

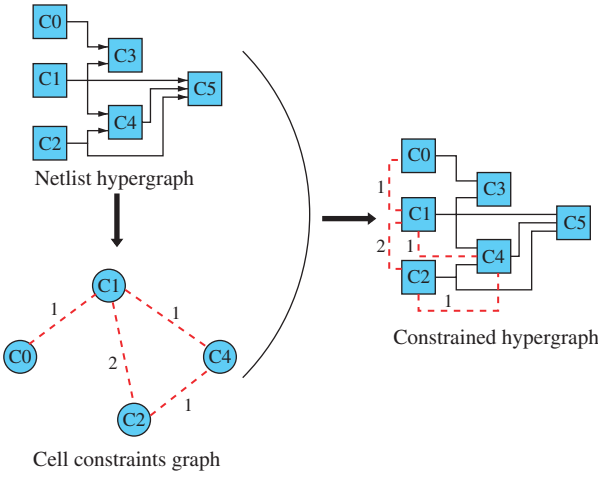

Figure 6. CCH: Cell constraints hypergraph.

It is worthwhile using the lowest level feedback link to connect a source to its destination, since it has an important impact on delay reduction. That is why the ACCG is constructed, ℓ_{up} corresponds to the lowest level where the source has to go up to reach its destination. Reducing the conflict between sources using the lowest level is beneficial for the first routing iteration. In fact, as will be explained in the next section, an iterative rip-up routing algorithm based on the congestion negotiation is used. An adjustable cost to each feedback is assigned. A lower level induces lower cost; consequently in the first routing iteration, signals will be routed using the lowest levels. Using the lowest levels to construct the ACCG has two advantages:

- fewer switches will be crossed to route signals
- a good initial solution for the iterative router exists: first iteration is run with the least number of resource conflicts

The construction of the ACCG is described in Algorithm 1.

Algorithm 1:

```
for each leaf cluster cl
  for each level l
    for each receiver rc of cl in level l
      for each leaf driver dr of rc
        //cl and dr both drive rc
        if No edge between cl and dr
          create edge e between cl and dr
        end if
        shift = ShiftCompute(cl,dr,rc)
        append shift to edge e
      end for
    end for
  end for
end for
```

Figure 7 presents the advanced cell constraints graph constructed from the placed netlist in figure 5. Since *cell*1 and *cell*2 drive a common destination, a directed edges pair is established between both vertices. The weight presented on each edge corresponds to the shift value: $(\ell 1_{up} - \ell 2_{up})$ *mod* 4 and $(\ell 2_{up} - \ell 1_{up})$ *mod* 4.

3.3.2 Simulated annealing strategy. A detailed placement consists of assigning a position for each cell inside its owner cluster. The objective is to reduce the number of resource conflicts. To compute this number, one takes each vertice in the ACCG

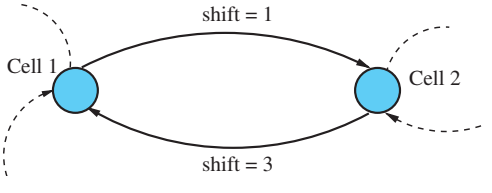

Figure 7. ACCG: advanced cell constraints graph.

and checks whether the condition of Lemma 1 is verified, if then the global cost function is incremented by 1. Computing this cost for a specific detailed placement is given by the following procedure.

Algorithm 2:

```
cost = 0
for each vertice v
  mark vertice v visited
  for each adjacent vertice adj of v
    if adj was not visited
      for each shift value
    (*) If conict(v,adj,shift)
          cost++
        end if
      end for
    end if
  end for
end for
```

The cost is updated incrementally in the sequel. To check whether there is a resource conflict (*), one must check the condition of Lemma 1. In order to do this, information about the first source position, the second source position (adjacent) and the forbidden shift value is needed. All the information is provided by the ACCG.

To find the best detailed placement combination, it is proposed to use an adaptive simulated annealing algorithm (Kirkpatrick *et al.* 1983). In this algorithm, the operating parameters are controled using statistical techniques (Aars *et al.* 1985).

Moves are randomly applied to the configuration and consist of assigning new positions to cells. First, an element to be moved is randomly chosen. Secondly, we randomly choose the new position inside the direct cluster owner and, if it is occupied, both elements positions are swapped. The cost function is updated incrementally by evaluating the incremental cost of the moved vertices and their adjacents. A hard windowing move restriction approach is adopted. A cell can only move inside its direct owner. This restriction is important to keep the partitioning result constant. In addition, by respecting this restriction, it is not necessary to update the ACCG since the common receivers and the levels to use to reach them always remain the same. This yields important run time reduction for the cost updating phase.

Moves that decrease the configuration cost are always accepted, while moves that increase the cost function are accepted with a probability that is directly related to the temperature and inversely related to how much the move impairs the system as a whole. As annealing proceeds, the temperature is slowly lowered.

4. Routing

The routing problem can be stated as assigning signals to routing resources, in order to successfully route all signals. This goal is difficult to achieve in the architecture, because of the lack of routing resources (depopulated switch boxes). In fact, the number of paths to reach a destination from a source is significantly reduced and

those paths depend on the location of cells and the number of levels in the architecture. Thus, signals will compete for the same resources and the challenge is to find a way to allocate resources allowing all signals to be routed. Despite this disadvantage, there is a great advantage in our architecture, since our unique path connecting an MSB to a logic block is predictable.

To route the architecture, a particular iterative rip-up algorithm based on the congestion negotiation called PathFinder (McMurchie and Ebeling 1995) is adopted. PathFinder was applied to the mesh architecture and adapted to MFPGA architecture. Since there is a unique downward path to reach a destination from an MSB, the breadth-first search in the detailed routing part has been eliminated.

Since the choice of the feedback imposes the path to use, the negotiation must be done on the choice of the feedback that leads to a path with less congestion. According to this remark, an adjustable cost is assigned to each feedback. The global router dynamically adjusts the congestion penalty for each feedback. During each iteration, individual routing resources may be used by more than one signal. The penalty to use shared resources is gradually increased, so that signals will negotiate effectively for resources. The implemented algorithm is described by the Algorithm 3.

Algorithm 3:

```
While shared resources exist
  /*global router*/
  Loop over all signals i
    Loop until all sinks tij are found
      Rip up branch Bij
      Find feedback j with lowest cost
      Bij <- j
      /*detailed router*/
      Loop until new tij is found
        Find next_wire
        Add next_wire to Bij
    End
  End
End
  /*backtrace*/
  Loop over nodes in Bij
  /*path from tij to si*/
    Update cost of j
  END
END
```

5. Timing analysis

This work is intended to evaluate the performance of MFPGA architectures in term of their functional speed. Thus, once an application has been completely placed and routed it is proposed to estimate the minimum feasible clock period to run it. Since there is a tree-connecting network, it is proposed to divide a path into several sub-paths. Each sub-path connects a source to a sink and consists in going from

a source up to a particular level and then down to the sink. The number of sub-paths depends on the number of levels. The first step consists of estimating delays on each sub-path, next compute the delay for each path composed of several sub-paths. This method enables the estimation of the clock frequency for applications implemented on MFPGA.

5.1 *Sub-paths delays evaluation*

A sub-path connects a source to a sink and crosses several MSBs. The number of sub-paths in the architecture is limited and depends on the number of levels. Consequently, given an architecture with n levels, one can isolate the n different sub-paths (symmetric structure). In figure 8, the 3 isolated sub-paths of an architecture containing 3 levels are shown. The SPICE circuit simulator is used to obtain highly accurate delay estimation in each sub-path. Each architecture is composed of combinational sub-paths that either start at a logic block (combinational/sequential) or an input pad *pi* and either end at a logic block (combinational/sequential) or an output pad *po*. To ensure proper circuit function, it is also necessary to take register setup-times t_{set} and sequential propagation delays d_{seq} into account (sometimes denoted as "clock-to-Q" delays). Classification of sub-paths and resulting delays is given below.

1. Combinational logic block → Combinational logic block
   $d(p) = d(switches)$
2. Combinational logic block → Output-pad
   $d(p) = d(switches) + d(po)$

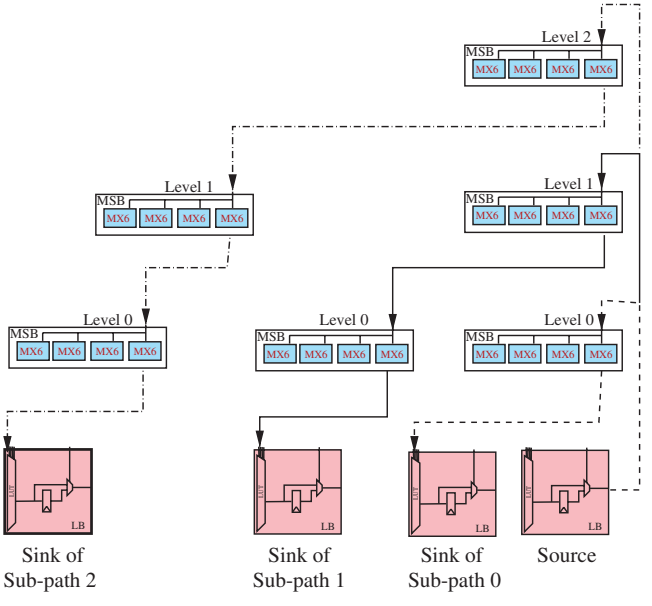

Figure 8. Sub-paths timing caracterisation.

3. Input-pad → Combinational logic block
   $d(p) = d(pi) + d(switches)$
4. Sequential logic block → Sequential logic block
   $d(p) = d_{seq} + d(switches) + t_{set}$
5. Sequential logic block → Combinational logic block
   $d(p) = d_{seq} + d(switches)$
6. Sequential logic block → Output-pad
   $d(p) = d_{seq} + d(switches) + d(po)$
7. Input-pad → Sequential logic block
   $d(p) = d(pi) + d(switches) + t_{set}$
8. Combinational logic block → Sequential logic block
   $d(p) = d(switches) + t_{set}$

Delays on sub-paths depend on the length of wires connecting MSB and logic blocks. These lengths are extracted from the routed MFPGA layout. Figure 9 shows the placed symmetric layout of a 4-levels MFPGA architecture (256 LBs). The basic tiles of the structure are:

- the LB that contains one multiplexer 16:1, one flip-flop and a bypass 2:1 multiplexer;
- the MSB that contains 4 multiplexers;
- the configuration memory blocks composed of 16 SRAM cells;
- the decoder for configuration memory adressing.

These basic tiles are duplicated at each level to construct the hierarchy recursively. Those tiles using a symmetric H″ plannig technique are abutted.

The MFPGA prototype is targeted to 0.13 CMOS process with 6 metal layers. This layout could be packed more tightly, since it has large tile areas composed of standard cells. With a Full-Custom technique, a more careful design of the interconnect coupled with the sub-clusters resources would decrease the total area and improve delay performances.

5.2 *Critical path extraction*

Once the circuit has been placed and routed, a direct graph called ''routing graph'' is obtained . This graph describes wires that will be used to connect logic block pins as described in the netlist according to the use made with architecture routing resources. Each wire and each logic block pin becomes a node in this ''routing graph'' and each

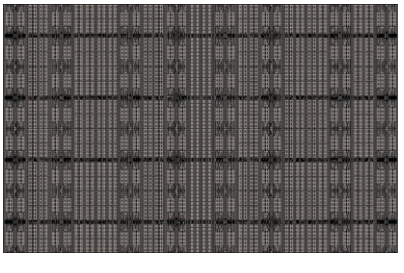

Figure 9. 4-levels symmetric MFPGA layout.

passing switch (inside the MSB) becomes a directed edge. Edges are also added between inputs of logic blocks and their outputs. Figure 10 shows a simple circuit implemented via 2-input LUTs and registers, and the corresponding "routing graph". On this graph, one can easily isolate different sub-paths through a depth-first traversal. Each sub-path is replaced by only one edge labelled with the sub-path delay.

Thus, a new direct acyclic graph called "timing graph" is obtained. In this graph, nodes represent the input pins and output pins of basic circuit elements, such as registers and LUTs. Register input pins are not linked to register output pins. Register outputs have no edges incident to them and register inputs have no edges leaving them (acyclic graph). Similarly, primary inputs (input pads) have no incident edges and primary outputs (out pads) have no exit edges. Each edge is labelled with the delay required to pass through circuit element or routing (sub-path delay). Figure 10 shows the obtained "timing graph" of the routed circuit.

One can determine the minimum required clock period with $O(n)$ computation for a "timing graph" with n nodes via a breadth-first traversal. This traversal begins at nodes with no incident edges (primary inputs and register outputs) and labels each with a signal arrival time, $T_{arrival}$, of 0. Each node which has incident edges from already labeled nodes is then labelled with its arrival time according to

$$T_{arrival}(i) = \max_{j \in fanin(i)} \{T_{arrival}(j) + delay(j, i)\}$$

where node i is the node being labelled, and $delay(j, i)$ is the delay value marked on the edge joining node j to node i. This procedure continues until every node in the graph has been labelled. Then the node with the largest arrival time, which will be always a primary output or a register input, defines the maximum delay, D_{max} (= minimum clock period), through the circuit. In figure 10, for example, the arrival time at node *Reg* is 5.5 *ns*, which is the largest arrival time, and hence the maximum circuit delay.

6. Experimental results

To evaluate architecture and tools performances, some of the MCNC benchmark circuits are placed and routed .

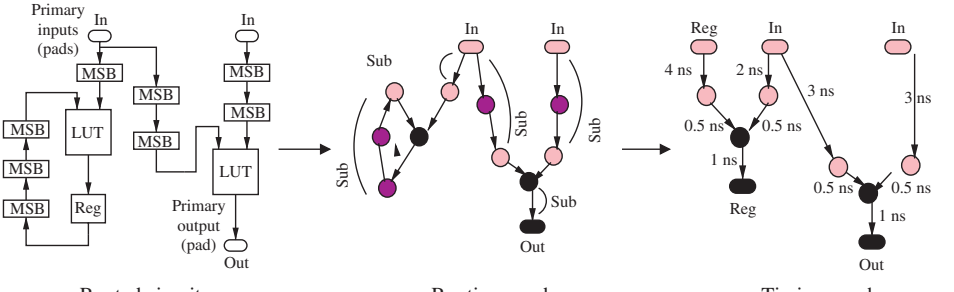

Figure 10. Timing graph modeling of a simple circuit.

6.1 *Density performances*

The same benchmark circuits are used to compare the switch and area requirements between MFPGA architecture and clustered mesh topology ($\sqrt{N}x\sqrt{N}$ clusters). The mesh architecture uses uniform routing with single-length segments and a subset switch box. Each cluster logic block contains four 4-LUTs. 10 inputs and 4 outputs appear on the 4 sides of the cluster. Both inputs and outputs are fully populated ($F_c = 1$). *IO* pads are fully populated too. A t-vpack is used (Marquart *et al.*1999) to construct clusters and the channel minimizing router VPR 4.3 (Betz *et al.* 1997) to route the mesh. VPR chooses the optimal size as well as the optimal channel width needed to place and route each benchmark.

Compare the areas of both architectures using successively a simple cost model based on routing switches count, and a more refined model that 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 of SRAMs and Multiplexers. The same cells library is used for both architectures. In table 1 it is shown that circuits on MFPGA using a smaller area than with mesh architecture can be implemented. The area reduction is about 17.5%. It is noticed that, in some cases (''tseng'' and ''alu4'' circuits), MFPGA is penalized by its low occupation.

6.2 *Speed performances*

It is clear from the previous comparison that the MFPGA architecture is more efficient in terms of area and this has a positive effect on circuit speed: the smaller the area, the shorter are the connecting wires. In addition, due to the multilevel hierarchy aspect of MFPGA architecture, logic blocks are partitioned between clusters to reduce external communications. This has an important impact on circuit speed improvement, since communication is faster. The speed of the MFPGA architecture to the Mesh was compared. The same circuits were implement the timing analysing tool was used for the MFPGA and the one proposed in VPR for the mesh (note: we applied a VPR timing-driven placement and routing).

Table 1. Area comparison: MFPGA vs mesh.

		Mesh			MFPGA			
Circuits	Size	$\sqrt{N}$	Switches number	Area ($\lambda^2 \times 10^3$)	Arch	Occup	Switches number	Area ($\lambda^2 \times 10^3$)
alu4	584	13	81926	207423	$4\times4\times4\times4\times4$	57	106496	299008
C5315	725	19	220004	555362	$4\times4\times4\times4\times4$	69	106496	299008
tseng	1047	17	210014	531058	$4\times4\times4\times4\times4\times2$	51	253952	679936
ex5p	1064	17	315084	785075	$4\times4\times4\times4\times4\times2$	51	253952	679936
apex4	1262	19	329894	822706	$4\times4\times4\times4\times4\times2$	61	253952	679936
dsip	1370	27	396504	1009107	$4\times4\times4\times4\times4\times2$	66	253952	679936
misex3	1397	20	353376	882555	$4\times4\times4\times4\times4\times2$	68	253952	679936
diffeq	1497	20	332324	836917	$4\times4\times4\times4\times4\times2$	73	253952	679936
bigkey	1707	27	337062	864003	$4\times4\times4\times4\times4\times2$	80	253952	679936
AVER	1183	20	286243	721578		64	221184	595285

Table 2. Speed Comparison (0.13 µm CMOS, 1.2 V)

Circuit	LUTs	Mesh	MFPGA
pcle	29	6.1	4.34
decod	32	5.36	3.56
cc	33	6.5	4.25
count	37	8.55	7.78
b9	61	10.77	6.98
i4	110	11.14	6.03
c2670	363	19.95	9.97
i9	471	15.67	10.90
alu4	584	20.9	11.26
C5315	725	24.63	15.17
Average	245	12.95	8.02

Timing results are presented in table 2. In this comparision, only small benches (< 1024 BLEs) were used. In fact, until now only architecture layouts (16, 64, 256 and 1024) have been generated and, as explained in § 5, layout information (wires lengths) is important for sub-paths delay caracterization. It is noted that MFPGA largely outperforms clustered mesh architecture (40%) in terms of speed despite timing driven placement and routing techniques having not yet been integrated.

7. Conclusion

The preliminary results show that good balancing of the LUT and the interconnect utilization reduces area compared with traditional mesh architectures. Thanks to the hierarchy aspect of the MFPGA there is a very good performance in terms of speed compared with the common mesh architecture.

The routing key of the proposed architecture is the upward network. Enhancing the routability needs to populate the upward network to increase the number of paths between sources and destinations. This can lead to area increasing, but can be compensated by applying Rent's rule to reduce the cluster signals bandwidth.

References

A. Aggarwal and D. Lewis, ''Routing architectures for hierarchical field programmable gate arrays'', *International Conference on Computer Design*, October 1994.

F.H.L. Aarts, F.M.J. Debont and E.H.A. Habers, ''Statistical cooling: A general approach to combinational optimization problems'', *Philips Journal of Research*, 4, pp. 193–226, 1985.

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

V. Betz and J. Rose, VPR: ''A new packing, placement and routing tool for FPGA research'', *International Workshop on FPGA*, pp. 213–222, 1997.

P. Guerrier and A. Greiner,''A generic architecture for on chip packet-switched interconnections'', *Design Automation and Test in Europe Conference*, Mars 2000, pp. 250–256.

G. Karypis and V. Kumar, ''Multilevel *k*-way hypergraph partitioning'', *Design Automation Conference*, pp. 343–348, June 1999.

S. Kirkpatrick, C.D. Gelatt and M.P. Hecchi, "Optimisation by simulated annealing," *Science*, 220, pp. 671–680, 1983.

Y. Lai and P. Wang, "Hierarchical interconnection structures for field programmable gate arrays", *IEEE Transactions on VLSI Systems*, June 1997, pp. 186–196.

C. Leiserson, "Fat-trees: universal networks for hardware efficient supercomputing", *IEEE Transactions on Computers*, pp. 892–901, 1985.

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 Field Programmable Gate Arrays*, February, pp. 37–46, 1999.

L. McMurchie and C. Ebeling, "PathFinder: a negotiation-based paerformance-driven router for FPGAs", *International Symposium on Field Programmable Gate Arrays*, February 1995, pp. 111–119.

W. Tsu *et al.*, "HSRA: High speed hierarchical synchronous reconfigurable array", *International Symposium on Field Programmable Gate Arrays*, February 1999, pp. 125–134.