# Multilevel Based Global Routing Algorithm for Hierarchical FPGA*

**Zhu Limin, Bian Jinan, Zhou Qiang, Hong Xianlong**

Department of Computer Science & Technology, Tsinghua University

**Abstract**

This paper presents an efficient global routing algorithm for a hierarchical interconnection architecture of FPGA. What is different from the traditional FPGA routing algorithm is that the proposed algorithm takes advantage of the hierarchical structure of this particular FPGA. We use a hierarchical tree as the routing resource representation of the corresponding interconnection architecture. In the routing phase, the global routing problem for each net is represented as a sub-tree determination problem. As soon as the location of each Logic Block is fixed, we can use a tree-growth-like algorithm to determine the sub-tree on the corresponding routing resource tree. The algorithm is very efficient and fast since the sub-tree is determined once we determined the position of each Logic Block. On the other hand, we can use this method to evaluate the routability of the corresponding placement results.

**Keywords**: FPGA, Hierarchical Architecture, Routability, Global Routing Algorithm

## 1. INTRODUCTION

The Field Programmable Gate Array (FPGA) is widely used for ASIC design and system prototyping. It reduces the design cost and implementation time. The traditional symmetrical array based FPGA [5] consists of Configurable Logic Block (CLB), Switch Box, Routing Segments and I/O blocks. Switch Box can be configured to connect Routing Segments into networks. I/O blocks provide the interface between the input/output pins and internal signal lines.

The key advantage of FPGA over the traditional gate array devices is the programmability of its logic function. As a result, the architecture of FPGA is far more complex. The logic and routing resources are limited in FPGA. Compared to traditional full-custom devices, FPGA suffers from low density and low performance. A signal in an FPGA connection path must pass through a lot of Switch Boxes. Since Switch Boxes take up extra chip area, an FPGA has lower density than conventional full-custom devices. Generally, the delay of the Switch Box is much larger than that of the wire. The delay of the signal path also increases due to the Switch Box interconnection.

As the development of the deep-micron technology, now we can build more transistors into a single chip. As a result, the designs become more and more complicated. The need to model the design quickly and efficiently becomes more and more important. On the other hand, the performance of an FPGA can be increased by reducing the number of Switch Boxes along a signal path. In addition, the density of an FPGA can be increased by reducing the total number of pass devices in a structure [1]. Based on these considerations, in this paper, we

*Supported by National Natural Science Foundation of China (NSFC) 90607001 and 607201066003

will describe the Hierarchical FPGA (HFPGA) interconnection structure presented in [1]. It has all of the previous advantage over the traditional Island-based FPGA. As a matter of convenience, we reproduced it as shown in Figure 1. This hierarchical structure of FPGA can dramatically reduce the amount of Switch Boxes along signal paths [1]. We can model a complicated design more easily and quickly on this architecture. In this paper, we will present a multilevel based global routing algorithm that takes advantage of the special characteristics of this hierarchical architecture. As the experimental result shows, the proposed algorithm is very efficient and fast, compared to the improved Maze routing algorithm in VPR.

This paper is organized as follows. In section II, some general routing algorithms for FPGA is discussed. Section III describes the new HFPGA architecture and its routing resource representation. The multilevel tree based global routing algorithm is presented in section IV. The experimental results and discussions are given in section V. Section VI concludes this paper.

## 2. FPGA ROUTING ALGORITHM SURVAY

Over the years, many algorithms have been proposed to perform routing on the FPGA. As the FPGA global routing problem is very similar to that of traditional metal- programmable gate-array (MPGA) or standard cell designs, many ASIC global routing techniques may be used for FPGA global routing. The early FPGA router CGE [9] and SEGA [10] adopted the global router LocusRoute [11] for standard cell designs. The main goal of CGE is to distribute the connections among the channels so that the maximum channel density is minimized. But by far, the most famous FPGA global routing algorithm, like the one used in Pathfinder [8] and VPR [3], is based on the negotiation-based global router [12] for standard cell designs.

Pathfinder uses an iterative algorithm that converges to a solution in which all signals are routed while achieving close to the optimal performance allowed by the placement. The router in VPR is based on the Pathfinder negotiated congestion algorithm. By gradually increasing the cost of oversubscribed routing resources, the algorithm forces nets with alternative routes to avoid using oversubscribed resources, leaving only the net that most needs a given resource behind.

In order to avoid the possible mismatch between global and detailed routing due to the difficulty of approximating all available routing resource in FPGA designs, several FPGA routers combine global and detailed routing in one step and produce good results. One of them is Tracer [4]. It use a simulated evolution based optimization technique to iteratively rip-up and reroute the nets violating the routing resource or timing constraints.

Almost all of these algorithms are based on the directed acyclic graph, which is the representation of routing resource for symmetric array based FPGA. These algorithms are all applied to the traditional island-style architecture. As the scale of the routing problem becomes bigger, the time complexity is becoming extremely larger. Sometimes we need an efficient method to quickly model a design and get a viable solution. In this paper, we present a multilevel tree based global routing algorithm which uses a hierarchical tree representation for the routing resource of hierarchical architecture of HFPGA. The experimental results show that this algorithm is very efficient and fast compared to the traditional FPGA routing algorithm since the algorithm is deterministic once the placement of the circuit is determined.

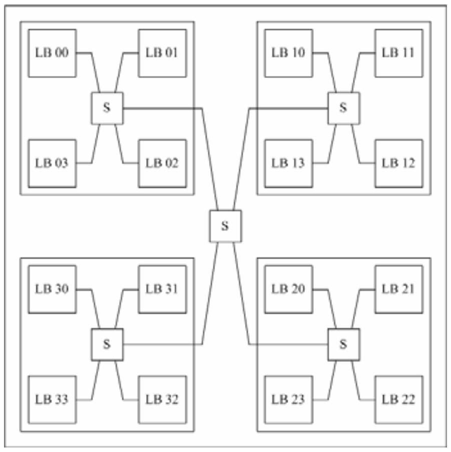

Fig. 1: The structure of an HFPGA

## 3. Hierarchical FPGA Architecture

### 3.1. Architecture Overview

The logic blocks in an HFPGA are connected in a hierarchical fashion. First, k logic blocks are connected to form a cluster with a switch box. Then k clusters are recursively connected together as a super-cluster. The structure of an HFPGA can be represented as a k-way tree. In [1], we know that k=4 is the optimal choice as for the density and performance of an HFPGA. In this paper, we will use k=4 as our default k value.

Figure 2 illustrate the corresponding 4-way tree of the hierarchical structure in figure 1. A leaf vertex of the tree represents the Logic Block and the other vertices correspond to the Switch Boxes, each of which has 4 children in this case. An edge in the tree represents a channel which consists of wire tracks.

In [2], we know that a connection path between two Logic Blocks in an HFPGA generally passes fewer switches than a path in the conventional FPGA. From [1], we see that the total number of switches in HFPGA is fewer compared with the island-style FPGA if they have same routability. The HFPGA architecture is kind of superior from the stand point of the routability and performance.

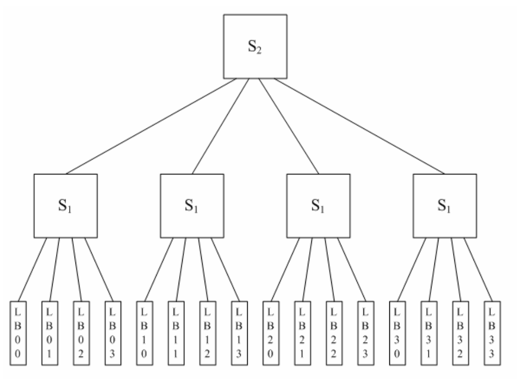

Fig. 2: The tree representation of an HFPGA

### 3.2. Routing Resource Tree of HFPGA

In order to model all the available routing resource in an FPGA, a routing resource graph is created as an abstract data representation to be used by the global and detailed routers in the conventional FPGA. In order to take advantage of special characteristics of the HFPGA, we model the routing resource of an HFPGA as a complete k-way tree. We will see that our efficient global routing algorithm is built on this tree. In this section, we will describe the building of the resource tree in detail.

### 3.3. Building of the Routing Resource Tree

As we mentioned earlier, the overall HFPGA architecture can be represented as a k-way complete tree. The leaf vertices are the Logic blocks while the other vertices are the Switch Boxes. Each edge represents the routing channel between Switch Boxes and Logic blocks (or Switch Boxes). The capacity of each edge represents the number of tracks in the corresponding routing channel. We can recursively build the routing resource tree using the logic blocks information and switch boxes information. After building the tree, we can use a quaternary number to uniquely identify the location of a logic block in a 4-way tree. Now we are ready to study various routing algorithm on the tree.

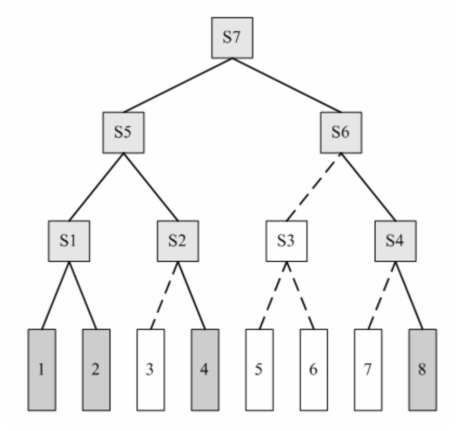

Fig. 4: the structure of a tree for a net

## 4. The proposed algorithm

From above discussion, we know that the interconnection structure in the HFPGA is very different from the traditional symmetric array based FPGA. The traditional graph based routing algorithms ignore the architecture characteristics of the HFPGA. In this paper, an efficient multi-level tree based global routing algorithm is proposed. As we know, the hierarchical interconnection architecture of the HFPGA corresponds to a k-way complete tree. The proposed algorithm takes advantage of the characteristic of the corresponding k-way complete tree to route all of the nets. This section describes the detailed implementation of this algorithm.

### 4.1. Constructing of Netlist Trees

During the partition and placement phases of FPGA design, the locations of the Logic Blocks in the netlist are fixed on the k-way routing resource tree. We all know that the netlist of the circuit in FPGA design is a Super-Graph where vertices represent the Logic blocks and super-edges represent the nets. A net is composed of multiple terminals which are electrically equal. Every net can be seen as a subtree mapped into the big routing resource tree. For example, if we need to connect Logic Block 1, 2 4 and 8 with a net in Fig 4, we must use the Switch Box S1, S2, S4, S5, S6 and S7 to route the net. The root vertex of this sub-tree is S7. We can see from Fig. 4 that once the locations of the Logic Blocks is fixed on the routing resource tree, the sub-tree of each net is determined in terms of the Switch Boxes and Logic Blocks. The algorithm for constructing the sub-tree for each net is listed as follows:

```
Algorithm: Construct Netlist Trees
For each net in the netlist
   iblk = block for source terminal of the net
   allocate a leaf node for iblk as a one-node tree
   For each sink terminal of the net
      Allocate a leaf node for this terminal block
      AddNode(node, tree)
   End for
End for

Function: AddNode(node, tree)
root = the root node of the tree ;
if the built tree covers node
   build the node on the path from root to node
else
   determine a new root of tree covering root and node
   build the node on the path from new root to node
   Build the node on the path from new root to old root
End if
```

Since the structure of the trees for each net is determined, the algorithm to construct the netlist trees is very efficient. The constructing of tree for each net is composed two processes: one is to initially construct a leaf node as a tree for the source terminal block of the net; the other is to iteratively add the leaf node for each sink terminal of the net to the tree, and add Switch Boxes for interconnection if needed. In this algorithm, this process is represented by the function AddNode. In this function, two different situations are processed based on the location of the new node. If the built tree covers the new node, that is, the new node is under the root of the built tree. For example in Fig. 4, suppose we have built the sub-tree for Logic Block 1 and Logic Block 4. When we want to add the new Logic Block 2, at this time, we only need to connect between the Logic Block 2 and Switch Block 1, since the Switch Block 1 and Switch Block 5 has been in the sub-tree. On the other hand, if the built tree doesn’t cover the new node, that is, the location of the new node is not under the root. At this time, we need to find a new root of a new tree, which both covers the root and the new node. This process is easy due to the fact that we represent the tree in quaternary numbers. We can shift the quaternary numbers for the location of each Logic Block to get the new root. After the new root is determined, we then need to connect the path between the new root and the old root, and the path between the new root and the new node. Then the root is updated. After processing each terminal of the net, the tree for the net is constructed. For example, in the Fig. 4, we assume the source terminal of the net is on LB1, while the other sink terminals are on LB2, 4 and 8 respectively. We first construct a leaf node for LB1, and then add a leaf node for LB4. In this process, the new root S5 is determined and S1 as well as S2 is inserted for interconnection between LB1 and LB4. When we add LB2 to the tree, we only need to connect between S1 and LB2, since the tree covers the node LB2. At last, we add node LB8 and S7 becomes the new root of the tree.

## 4.2. Global Routing

As we know, the global routing determines the coarse routing topology of each net in terms of channels. Since we construct the sub-tree for each net, the channels traversed by each net are determined. In this phase, we project the constructed subtree of each net onto the routing resource tree by a specified order. If we can route all the nets in the netlist without and resource conflict, the global routing is successful. On the contrary, if we run into some conflicts, then we need to evaluate the global routing in terms of the conflicts, and rip-up the global routing and re-place and reroute the circuit.

First, we break each net into some track segments in channel. Then we project each track segment in the channel by the order of the level of the segment. We decided to route the netlist in bottom-up fashion, as the number of the tracks needed to be routed is bigger, so is the conflicts. The whole process is very efficient and fast because we have constructed the sub-tree structure for each net in the first place, the only thing we do in this phase is route all the net in the bottom-up order. It is determinant once the sub-tree structure is available. After all of these is finished, we could easily evaluate the routing results.

## 4.3. Evaluating routing

We defined the evaluation of the routing results in terms of the resource conflicts ever encountered in the process of the global routing. We formulate as follows:

$$Conf(routing) = \frac{\sum_i N_{conf}(net_i)}{\sum_i N_{trks}(net_i)} \qquad (1)$$

In the equation 1, the Conflict for the result routing is defined to be the ratio between the sum of the number of conflicts and the sum of the number of tracks for each net. If the Conflict is zero, then we consider the routing successful. We can also use this equation in the placement phase as a guide.

### 4.4. Time Complexity Analysis

We first analyze the time complexity of the routing sub-tree construction algorithm. For each net of fan-out k, we need to invoke the function AddNode k-1 times. The AddNode then find a proper location to put the new node. From the Fig. 4, we know that the maximum path for the AddNode to look up is proportional to the levels of the routing resource tree. On the other hand, the levels of the routing resource tree are determined by the total number of Logic Blocks, i.e. $\lceil \log N \rceil$, in which N represent the number of the logic blocks. Moreover the routing path is determined. So compared to the improved Maze router in VPR, it is very efficient and fast.

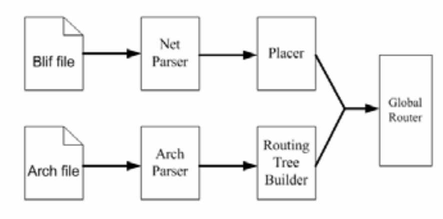

Fig. 6: Flow for HFPGA global routing

## 5. EXPERIMENTAL RESULTS

We have implemented the basic infrastructure and algorithm in C++ language. The implement flow is as follows (Fig. 6). Like in VPR [3], we use an architecture file to describe the architecture of HFPGA, where we put together architecture related parameters discussed in section III. After parsing the architecture file, we use a Fiduccia-Mattheyses [6] based partition algorithm to partition the netlist. Iteratively, we partition the netlist into four sub-circuits, each of which is randomly placed on each sub-tree. It is in this period that the location of each Logic Block in the HFPGA is determined.

Tab. 1: Experimental results for MCNC benchmark

| Circuits | #clb | #Net | T/s | T(Maze)/s | Improve |
|---|---|---|---|---|---|
| e64-4lut | 194 | 339 | 0.001 | 0.008 | 8 |
| ex5p | 1064 | 1072 | 0.015 | 0.255 | 17 |
| apex4 | 1262 | 1271 | 0.015 | 0.245 | 16.3 |
| misex3 | 1397 | 1411 | 0.015 | 0.195 | 13 |
| alu4 | 1522 | 1536 | 0.016 | 0.195 | 12.2 |
| des | 1591 | 1847 | 0.015 | 0.115 | 9.6 |
| seq | 1750 | 1791 | 0.016 | 0.250 | 15.6 |
| apex2 | 1878 | 1916 | 0.015 | 0.245 | 16.3 |
| spla | 3690 | 3706 | 0.031 | 0.475 | 15.3 |
| pdc | 4575 | 4591 | 1.766 | 6.656 | 3.8 |
| ex1010 | 4598 | 4608 | 3.781 | 7.262 | 1.9 |
| Aver | | | 0.51 | 1.445 | 11.7 |

On the other hand, an architecture file parser is used to parse the architecture file to get the needed parameters, which are applied to the routing tree builder to build a k-way hierarchical tree. At last, the global routing algorithm is applied to route the netlist on the built routing resource tree. The experiments have been carried out on the 11 large MCNC benchmark circuits. The characteristics of the circuits and their related experimental results are presented in Table 1.

## 6. CONCLUSION AND FUTURE WORKS

In this paper, we present an efficient multilevel tree based global routing algorithm for the hierarchical architecture of FPGA. We take advantage of the hierarchical characteristic of HFPGA to make the global routing extremely efficient and fast. We use a hierarchical tree to represent the routing resource of HFPGA. In the routing phase, the global routing problem for each net is represented as a sub-tree determination problem. As soon as the location of each Logic Block is fixed, we can use a tree-growth-like algorithm to determine the sub-tree on the corresponding routing resource tree. As the experimental results show, the proposed algorithm is very efficient and fast. In the future work, we are planning to combine the placement phase and global routing phase together to study the relationship between the placement and routing.

## 7. References

[1] Yen-Tai Lai and Ping-Tsung Wang, “Hierarchical Interconnection Structures for Field Programmable Gate Arrays” IEEE Trans. On VLSI Systems, Vol.5, No. 2, June 1997

[2] Mingjie Lin, Abbas El Gamal "TORCH: a design tool for routing channel segmentation in FPGAs" , Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays, Feb. 2008 International Symposium on Circuits and Systems.

[3] Vaughn Betz and Jonathan Rose, “VPR: A New Packing, Placement and Routing Tool for FPGA Research”, 1997 International Workshop On Field Programmable Logic and Application.

[4] Yuh-Sheng Lee and Allen C.-H. Wu, “A Performance and Routability-Driven Router for FPGA’s Considering Path Delays” IEEE Trans. On CAD of Integrated Circuits and Systems, vol. 16, no. 2, Feb 1997.

[5] Siva Nageswara Rao Borra, Annamalai Muthukaruppan, PS. Suresh, PV. Kamakoti "A novel approach to the placement and routing problems for field programmable gate arrays" Applied Soft Computing, Vol 7 no 1, Jan. 2007.

[6] C. M. Fiduccia and R. M. Mattheyses, “A linear-time heuristics for improving network partitions” Proceeding of the 19th Design Automation Conference, pp. 175-181, 1982.

[7] Deming Chen, Jason Cong and Peichen Pan, “FPGA Design Automation: A Survey” Foundations and Trends in EDA Vol. 1, No 3 (Nov. 2006) 195-330.

[8] Larry McMurchie and Carl Ebeling, “PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs” Proc of the Third International ACM Symposium on FPGA’95.

[9] S. Brown, J. Rose, and Z. G. Vranesic, “A detailed router for field programmable gate arrays”, IEEE Trans. Computer-Aided Design, vol. 11, no. 5, pp. 620-628, May 1992.

[10] G. G. Lemieux and S. D. Brown, “A detailed routing algorithm for allocating wire segments in field-programmable gate arrays”, in Proc. ACM/SIGDA Physical Design Workshop, 1993, pp. 215-226

[11] J. Rose, “Parallel global routing for standard cells. IEEE Trans. On Computer Aided Design of integrated Circuits and Systems, 9(10): 1085-1095, October 1990.

[12] R. Nair. “A simple yet effective technique for global wiring”, IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, CAD-6(6): 165-172, March 1987.