ROUTING FOR FPGA CIRCUITS

Zoltan Baruch[1], Octavian Creţ[2], Kalman Pusztai[3]

[1]*Technical University of Cluj-Napoca, Romania, E-mail: Zoltan.Baruch@cs.utcluj.ro*
[2]*Technical University of Cluj-Napoca, Romania, E-mail: Octavian.Cret@cs.utcluj.ro*
[3]*Technical University of Cluj-Napoca, Romania, E-mail: Kalman.Pusztai@cs.utcluj.ro*

Abstract - In this paper we present a specific routing algorithm for the *Atmel 6002* FPGA circuit. We define a specific cost function and a specific way of handling the connectivity list, sorting it by two main criteria: the number of stored links and the competition's cost. We implemented a routing algorithm which simultaneously treats the global and local routing. One of the advantages of considering the global routing and the detailed routing simultaneously is that the preliminary estimation of the global routing can be immediately and appropriately corrected. The algorithm may consider the side effects of the routing decisions made for one connection on the others, thus resolving the routing conflicts. According to the sorting of the connection list, we can have two kinds of optimizations: *area optimization* and *speed optimization*.

Key Words: Digital Design, FPGA, Routing.

1. INTRODUCTION

1.1 The FPGA Design Process

Field-Programmable Gate Arrays (FPGA's) are flexible circuits that can be easily reconfigured by the designer, reducing considerably the design cycle [5]. There are many commercial FPGA's. The RAM-based FPGA's, such as *Xilinx*'s *XC3000* and *XC4000*, or *Atmel*'s *6000* series [6], are a widely used class of them. The architecture of a RAM-based FPGA consists of an array of user configurable logic blocks, and a set of programmable interconnection resources used for routing [4]. Each logic block implements a part of the design logic, and the routing resources are used to interconnect the logic blocks.

In computer-aided logic design for FPGA circuits, the main operations to be performed by the software are the following:

- *Generating an internal representation*. The design specification in a hardware description language (HDL) must be compiled into an internal representation (usually a graph), which will be used by all the subsequent programs.
- *Technology mapping*. The intermediate form is adapted for the logic blocks inside the FPGA, considering the restrictions introduced by their architecture (number of inputs, number and type of the functions inside each block).
- *Placement*. Placement is the operation of assigning each vertex of the Boolean network to a specific logic block in the FPGA device. Every vertex of logical network is assigned to a logic block of the FPGA circuit.

- *Routing*. The routing achieves the interconnection of the logic blocks. This operation is usually performed in two steps. In the *global routing* step, interconnection paths at the global level are chosen, according to certain restrictions which are effective at the global level. In the *detailed routing*, the starting point is the result of the global routing, the goal being to establish the routing paths at the detailed level. The result of this step is a list of interconnection segments for each group of terminals.

In this paper we focus on the routing phase of FPGA design, particularly for the *Atmel 6002* circuit. We specify a routing algorithm which simultaneously performs the global and local routing. The advantage of this algorithm is that the preliminary estimation of the global routing can be immediately and appropriately corrected. The algorithm may consider the side effects of the routing decisions made for one connection on the others, thus resolving the routing conflicts. We can have two kinds of optimizations: *area optimization* and *speed optimization*.

1.2 Previous Work

There are only a few published routers for RAM-based FPGA's: *CGE* [3], *SEGA* [2] and *TRACER-fpga* [1].

CGE (Coarse Graph Expansion) [3] first uses a global routing to decompose each net into a number of two-terminal connections. Its primary goal is to distribute the connections among the channels in order to balance the channel densities. It then chooses for each two-terminal connection exact wiring segments to implement the path assigned during the global routing. A cost function is used to iteratively select among all exact paths the best one. This iteration halts when no more uncompleted exact paths are left.

SEGA (SEGment Allocator) [2] is intended for FPGA's with variable-length wiring segments. It addresses the allocation of wiring segments to connections with the goal to match the length of the wiring segments to the length of the connections. *SEGA* uses the same strategy as *CGE*; the main difference comes from the cost function.

TRACER-fpga [1] consists of two stages: initial router, and rip-up and rerouter. During the first stage, nets are routed sequentially and independently of one another, ignoring the existence of any previously routed nets. Inevitably, there will be conflicts over the usage of routing resources among nets. During the second stage, conflicts are resolved iteratively. Within an iteration, some nets are ripped-up and rerouted. The selection of nets for ripping-up is guided by a simulated evolution-based optimization technique. The rerouting is done with the expansion router, except that the presence of other already routed nets is no longer ignored.

2. GENERAL APPROACH OF THE ROUTING PROBLEM

The solution to the general routing problem usually requires a *division strategy*. Then the routing can be solved in three steps [3]:

1. *Partitioning* the routing resources.
2. Using a *global router* for assigning a set of routing areas to each net, creating a new set of restrictions.
3. Using a *local router* for selecting the specific wire segments and the routing switches for each connection, inside the restrictions set by the global router.

This approach has the advantage that each routing tool can solve better a little part of the routing problem. To be more specific, while the global router does not require to be related to the wire segment or the routing switches allocation, it can focus on more global points, balancing the use of the routing channels.

Similarly, having a reduced number of available routing alternatives for each connection (because of the restrictions imposed by the global router), the detailed router can focus on the effective connection implementation.

Placement and routing are interdependent because the FPGA cells can be used both for implementing logic functions and for routing. The relation between placement and routing creates a number of compromises, affecting the circuit speed. For example, the placement of one cell may interfere with the routing by locking the cell, making the net unroutable. Generally, because the number of logic cells is higher than the number of buses, on small distances the cells can be used instead of the buses. The placement interconnects adjacent or neighbor locations, avoiding the routing delays and facilitating simultaneous routing of the signals.

It is preferable that bus resources are saved for routing long distance signals (more than five cells) in the FPGA array. When a number of signals contain entries for more than one function, grouping the signals permits their parallel routing.

3. THE ROUTING ALGORITHM

The first step of the routing program consists of constructing a graph, and then selecting the specific routing segments for each graph. Therefore the allocation of routing resources is strongly dependent of the path chosen by the global router. Given that the FPGA device used here is not of high structural complexity (it includes only local buses, express buses, repeaters and logic cells), the chosen routing algorithm includes the detailed routing inside the global routing at each hierarchical level.

After the global routing, the connections assigned to each sub-channel are known. If the detailed router fails to route all the connections assigned to a sub-channel, then the channel's capacity is correspondingly reduced and the global routing at this level is redone. The advantage of this approach is that the preliminary estimation in the global routing can be corrected immediately.

It is recommended for the algorithm to reserve the bus resources for signals that go over longer distances (more than five cells), so that for these signals the advantages of the express bus become visible. The express buses are not directly connected to the cells, thus they have small capacities, are faster than the local buses, and it is indicated to use them as often as possible for increasing design performance. Also, by using an express bus instead of the local bus, the local bus is released for other necessary routings.

Substituting a local bus by an express bus is not possible in the following situations [6]:

- when directly connecting two cells
- when using a bidirectional signal
- when making 90° turns

For increasing design performance, it is indicated to limit the number of local bus segments which carry a signal and to go beyond the limit of the repeater only if it is necessary. Branching the express bus signal to the local bus at each repeater may be beneficial when using more than 8 signals or when the signal passes over more than one repeater. Figure 1 presents an example of ramification.

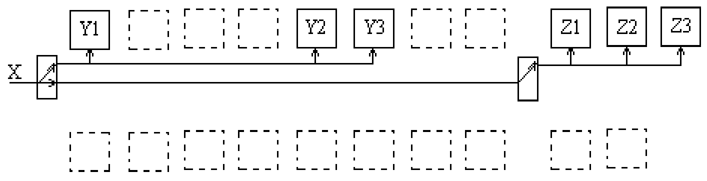

Figure 1. Example of signal ramification inside the FPGA structure.

Signal X is routed over the express bus and the branching to the repeaters of the local bus segment are leading to the Z and Y cells. If signal X would have been routed over the local bus to the Y1, Y2 and Y3 cells and over the repeaters (local bus to local bus) to the Z1, Z2 and Z3 cells, the load of the Y cells would have affected the speed of the signals which are branched to the Z cells.

Another key problem of the routing is the selection of the connection. Where two or more connections pass over a common routing channel, there may appear competitions for the routing resources in that channel. Because of the limited connectivity in the *Atmel 6002* FPGA, it is essential to resolve these conflicts.

The main problem of FPGA routing is that the choice made for one connection may block another connection [3]. Figure 2 shows two positions of the same section of an FPGA device. Each section offers routing options for either A or B connection. In the figure, the logic cell is denoted by L, the connectivity points by ×, the wire segment (local bus) by a solid line and the possible routing by a dashed line.

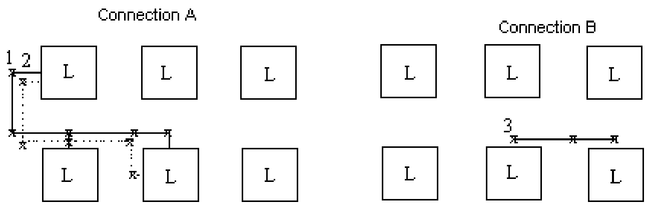

Figure 2. Routing conflicts.

Let us assume that the router first performs the connection A. If the wire segment number 1 is chosen for A, then connection B cannot be routed because B relies on the single left option, that is the wire segment number 1. The correct solution for the router is to select the wire segment number 2 for connection A; then connection B can also be routed. This is a simple example which illustrates the essence of the problem which appears because of the limited routing resources inside the FPGA device.

The algorithm cannot consider all the connecting possibilities inside the FPGA in a single phase because it has to save memory and execution time. This is the reason why it uses an iterative approach. In the first step it considers only those possible paths for a connection which correspond to the minimal cost (C_d) necessary to the algorithm to find out connecting paths. If these paths fail (are in conflict with the connections already routed), the algorithm continues its search starting with this failed path cost.

The routing algorithm

1. Build the connectivity graph based on the FPGA device structure. Divide the network in two-point connections, building in this way the routing graph.

```
2. Find the minimal distance of the path over the routing channels for each connection
   which could not be directly linked or broken in direct links, and with this minimal
   distance store all the linking alternatives in a connection list.
3. while (the connection list is not empty) do
          Sort the connection list by the number of linking alternatives;
          Select the connection from the head of the list;
          Sort the possible paths by $S_c$;
          if (the paths list is not empty) do
              Route the selected connection, using the minimal cost path;
              Mark the connection in the connectivity graph;
              Find all the paths which are in conflict with the selected path (for
                        example all the paths which use one of the connectivity points
                        of the selected path) and delete them;
          else do
              Reroute the selected connection starting from the connection's distance;
              if (it cannot be routed)
                      Mark the circuit as unroutable;
              endif
          endif
   endwhile
```

In the first step, the connectivity graph is built on the *Atmel 6002* FPGA device structure, and the routing graph is built from the results obtained after the technology mapping and placement steps. By *direct links* we denote here those links which do not imply the use of any bus.

In the second step, the connection list contains all the connections to be routed with the respective linking paths, corresponding to the shortest distance possible. There may exist more than one routing possibility with the same cost. This minimal distance is used as a cost which is computed according to the FPGA structure. The algorithm tries to break the connections in direct links if it does not have to pass over more than five cells, using the fact that the FPGA cells can also be used for routing.

In the third step, the algorithm sorts the connection list by two criteria:

- the number of stored links
- the cost of the competition

Each wire segment has two associated costs: the *distance cost* (C_d), which reflects the routing delays associated with the wire segment, and the *competition cost* (C_c), which counts the links' competition to the same wire segment.

Each path in the connection list has also two associated costs: the *sum of the distances cost* (S_d) of the wire segments in the path, and the *sum of the competition cost* (S_c) of the wire segments in the path.

The connection list is sorted by the number of stored links to determine what connection is essential. The essential connection must be identified for making the selection for a connection which has a minimal number of S_d-cost connecting alternatives.

The algorithm first sorts the connection list by the C_d cost (also because it tries to make an area and speed optimization). By first choosing the connections with minimal S_d cost, it forces the long lines to choose the express buses, which are much faster. From all possible connections, the one with minimal S_c cost is chosen.

At the routing step, the algorithm selects the linking alternatives with minimal S_c cost. At the rerouting step, it tries to find other possible paths starting from the minimal distance, based on the updated connectivity graph. In the connectivity graph, at this step, the connectivity points used by the connections already routed are marked. The driving cell (the one which generates the signal on the connection) is also marked in the connectivity graph, because we need signal branching in the local bus to become possible. Therefore at the rerouting phase the linking alternatives starting from a connectivity point which is used for other connections are already taken into consideration (but which contains the necessary signal for the connection which is currently rerouted).

With this algorithm other architectures can also be routed, if we isolate step 1 which is FPGA device structure-dependent.

4. CONCLUSIONS

Efficiently solving the routing problem is essential for any CAD system for designing with reconfigurable FPGA logic devices. The complexity of the problem is increased by the fact that routing and placing are strongly inter-dependent. Any routing algorithm is also dependent on the internal structure of the FPGA device.

In this paper we examined the general structure of any routing algorithm, and we presented a specific algorithm for the *Atmel 6002* FPGA device. We defined a specific *cost function* and a specific way of handling the connectivity list, sorting it by two main criteria: *the number of stored links* and *the competition's cost*. The competition may occur during routing, when many connections compete for a single routing channel. The existence of a limited number of routing channels inside the FPGA device increases also the complexity of the problem, so that *rerouting* may become necessary at one point.

The advantage of this algorithm is that the preliminary estimation of the global routing can be immediately and appropriately corrected. The algorithm may consider the side effects of the routing decisions made for one connection on the others, thus resolving the routing conflicts. According to the sorting of the connection list, we can have two kinds of optimizations: *area optimization* (when sorting by the sum of distance cost) and *speed optimization* (when sorting by the sum of competition cost). The algorithm is implemented in C programming language, as part of a CAD system for FPGA circuits.

5. REFERENCES

[1] C.-D. Chen, Y.-S. Lee, A. C.-H. Wu, Y.-L. Lin [1995]: “TRACER-fpga: A Router for RAM-based FPGA’s”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, No. 3, pp. 371-374.

[2] G. G. Lemieux, S. D. Brown [1993], “A Detailed Routing Algorithm for Allocating Wire Segments in Field-Programmable Gate Arrays”, *Proceedings of ACM/SIGDA Physical Design Workshop*, Lake Arrowhead, CA, pp. 215-226.

[3] S. D. Brown [1992], *Routing Algorithms and Architectures for Field-Programmable Gate Arrays*, Ph.D. Thesis, University of Toronto, Canada.

[4] S. M. Trimberger [1994], *Field-Programmable Gate Array Technology*, Kluwer Academic, Boston MA.

[5] Y. Sun, T-C. Wang, C. K. Wong, C. L. Liu [1997], “Routing for Symmetric FPGA’s and FPIC’s”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 16, No. 1, pp. 20-31.

[6] *Configurable Logic. PLD, FPGA, Gate Array*. Data Book, Atmel Corp., 1995.