ACM/SIGDA Workshop on Field-Programmable Gate Arrays, Berkeley, February, 1992.

Routing-directed Placement for the TRIPTYCH FPGA

Elizabeth A. Walkup, Scott Hauck, Gaetano Borriello, Carl Ebeling

Department of Computer Science and Engineering, FR-35
University of Washington
Seattle, WA 98195

Abstract

Currently, FPGAs either divorce the issues of placement and routing by providing logic and interconnect as separate resources, or ignore the issue of routing by targeting applications that use only nearest-neighbor communication. Triptych is a new FPGA architecture that attempts to support efficient implementation of a wide range of circuits by blending the logic and interconnect resources. This allows the physical structure of the logic array to more closely match the structure of logic functions, thereby providing an efficient substrate (in terms of both area and speed) in which to implement these functions. Full utilization of this architecture requires an integrated approach to the mapping process which consists of covering, placement, and routing. While our primary goal in this work is to develop effective mapping tools for the Triptych architecture, more tightly-coupled placement and routing will be an important approach for both general-purpose and domain-specific FPGAs. In this paper, we demonstrate that a routing-directed placement algorithm for Triptych should lead to cost-effective mappings when compared to other FPGAs.

1. Introduction

Current general-purpose FPGAs partition the chip into a set of logic blocks interconnected by programmable wiring in the form of wiring channels or a cross-bar matrix. FPGA mapping begins by dividing the circuit into pieces of sequential logic (*logic nodes*), each of which can be implemented or *covered* by one logic block. A netlist indicating how the logic nodes are connected maintains the functionality of the original circuit. The circuit is then realized by *placing* each logic node in a logic block and *routing* the inputs and outputs of the logic blocks together according to the netlist. Clear separation between the logic and interconnection resources is attractive because the covering, placement, and routing decisions are highly decoupled. The price of this separation is the large area and delay cost incurred by the flexible interconnection needed to support arbitrary routing requirements. This leads to architectures like Xilinx [Carter86], where the routing resources consume more than 90% of the chip area. Even so, the largest Xilinx FPGA (3090) seldom achieves more than 50% logic block utilization for random logic. A lack of interconnect resources also leads to decreased performance as critical paths are forced into more circuitous routes.

Domain-specific FPGAs like the Algotronix CAL1024 (CAL) and the Concurrent Logic CFA6000 (CFA) increase the chip area devoted to logic by reducing routing to nearest-neighbor communication [Algotronix91, Concurrent91]. The result is that these architectures are restricted to highly pipelined dataflow applications, for which they are more efficient than general-purpose FPGAs. Implementing circuits using these FPGAs requires close attention to routing during placement, which is generally accomplished by manual placement and routing of regular structures or using specialized tools optimized to a specific circuit domain [Kean89]. Some research has been done on integrating placement and routing for domain-specific FPGAs but this work is very preliminary [Beetem91].

It is clear that domain-specific FPGAs can benefit greatly from an integrated approach to covering, placement, and routing. But we believe it is also true that general-purpose FPGAs can take advantage of the increased flexibility offered by this approach. Although it is appealing to view routing and placement as a separate processes, poor placement can make routing impossible in FPGAs; therefore, this separation requires a large amount of highly flexible interconnect to make routing feasible.

We have taken this integration one step further to FPGA design itself by defining a new architecture with more finely blended logic and routing resources. Triptych is structured so that the physical layout of a mapped circuit can follow its logical structure, reducing the need for extensive routing resources and improving circuit delay properties. Taking full advantage of this less flexible but more efficient structure will require integrated covering, placement, and routing. In this paper, we discuss the effects of Triptych's architecture upon the mapping process and describe our work to realize CAD tools for these tasks.

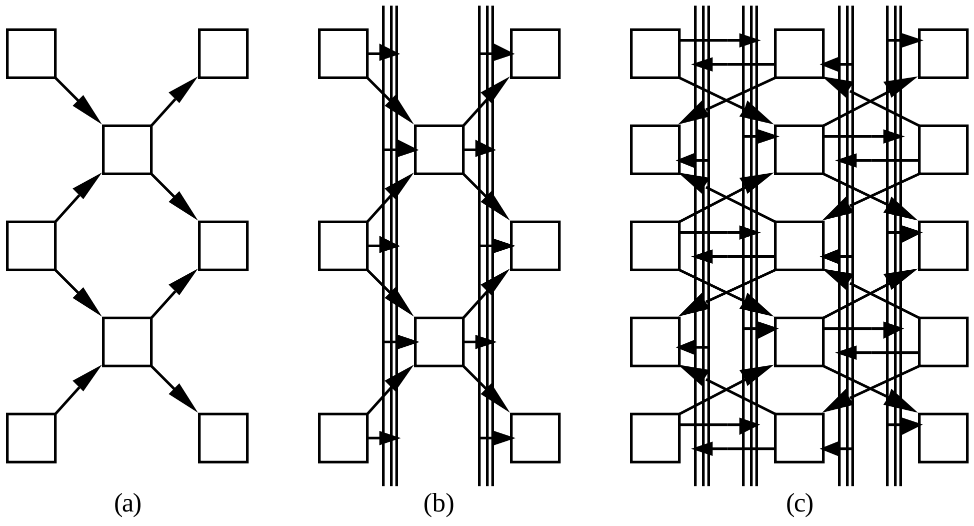

Figure 1. The overall structure of the Triptych FPGA shown in a progression of steps. The basic fanin/fanout structure on the left is augmented with segmented routing channels attached to a third RLB input and output. The structure on the right is obtained by merging two copies of the middle structure, with data flowing in opposite directions in the two copies. Not shown are the connections between the two copies, which permit internal feedback.

2. The Triptych Architecture

The signal flow in a typical multi-level combinational logic circuit can be viewed from an input as a fanout tree to those outputs that the input affects or from an output as a fanin tree from those inputs it depends upon. The Triptych architecture [Ebeling91] emulates this fanin/fanout tree form by arranging RLBs (routing and logic blocks) into columns, with each RLB having short, hard-wired connections to its nearest neighbors in adjacent columns (see Figure 1a). The basic structure is augmented with segmented routing channels between the columns which facilitate fanouts greater than are possible in the basic array (see Figure 1b). Finally, two copies of the array, flowing in opposite directions, are overlaid (see Figure 1c). Connections between the planes exist at the crossover points of the short diagonal wires. It is clear that this array does not allow arbitrary point-to-point routing like that associated with Xilinx and Actel FPGAs [Xilinx91, Actel91]. However, we claim that this array matches the form of many circuits, and that a mapping strategy that takes this structure into account can produce routeable implementations. Section 4 outlines the approach we are taking on integrating these routing limitations into the placement problem.

A Triptych RLB (shown in Figure 2) has three inputs and three outputs and may perform an arbitrary logic function of the three inputs, with the result optionally held by a master/slave D-latch. Only one input and one output can be connected to the vertical wires; the other two must be on the local diagonal interconnect.

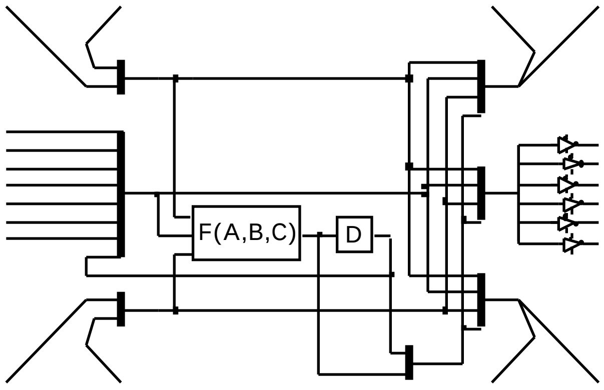

Figure 2. Triptych RLB design. The RLB consists of: 3 multiplexers for the inputs, a 3-input function block, a master/slave D-latch, a selector for the latched or unlatched result of the function, and 3 multiplexers for the outputs.

Local high-speed communication between neighboring RLBs is achieved through the "diagonal" connections. The detailed structure of the diagonals is shown in Figure 3. They allow RLBs to send outputs to the RLBs immediately above and below them, which flow in the opposite direction, and to the two closest RLBs in the next column that flow in the same direction. Diagonals are important for two reasons: they permit the construction of multi-level functions of more than three inputs without the speed penalty of general-purpose interconnect, and they allow signal flow to change direction both so that circuits can be more tightly packed and signals can be fed back when implementing sequential logic.

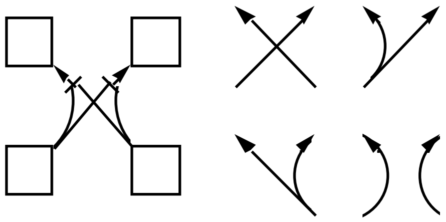

Figure 3. Schematic view of a pair of diagonals and the routing combinations they allow (implemented by a multiplexer at each diagonal input).

Each RLB is capable of performing both function calculation and routing tasks simultaneously, which leads to several different uses of the RLB some of which are shown in Figure 4. Since the Triptych FPGA has no global routing for moving signals horizontally, there is a heavy reliance on unused RLBs and unused portions of RLBs to perform these routing tasks. Routing in the Triptych array takes three forms: horizontally from one column to the next through the RLBs (by selecting an input to be routed to an output), diagonally through short wires to neighbors (for local connections), and vertically through segmented channels between columns of RLBs (for more global interconnect capabilities). The uses shown are, in order: routing only, routing with fanout, function block utilization with fanout, function block usage with continued routing of function inputs, and two-input function with disjoint routing.

For longer range connections and large fanout nodes, the segmented channels are used. They are implemented as a set of segmented "channel wires" between adjacent columns that connect middle outputs of RLBs to the middle inputs of RLBs flowing in the same direction in the next column. This flexibility leads to slower signal propagation, and speed-critical designs will avoid using the vertical channels along critical paths. There are 7 tracks in a vertical channel, with 6 handling inter-cell RLB routing and a seventh to carry an external input. The 6 inter-cell tracks are broken up into two tracks each of 8, 16, and 32 RLB high segments. In the current layout, Triptych measures 32 RLBs high by 8 RLBs wide.

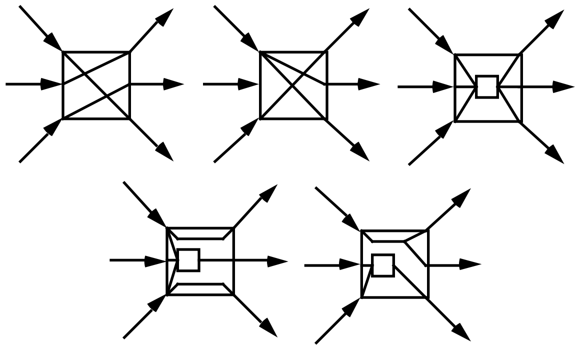

Figure 4. Five typical uses of Triptych routing and logic blocks (RLBs).

To demonstrate Triptych's viability we present Table 1, which displays normalized measures of the geometric mean areas and delays for three FPGAs relative to Triptych. For this normalized measure, the cost of using a single cell is considered to be the ratio of the cell's area to the area required by a single programming bit, a metric which among other things normalizes cost to a standard process. All circuits were synthesized and covered using MIS-FPGA and were hand mapped and optimized for area only. Area figures for Xilinx are lower bounds as only function covering was performed and it was assumed that during placement no cells would be unreachable. Xilinx routing delays were calculated via a delay model presented in [Schlag91]. The circuits include three random logic benchmark FSMs and 8 to 10 circuits each from arithmetic, systolic, and linear- and exponential-growth bit-parallel examples. For full details, please see [Hauck92].

	Triptych factor	CFA factor	CAL factor	Xilinx factor
Area	1.0	2.06	1.17	3.53
Delay	1.0	1.74	1.73	0.99

Table 1. Table of area and delay comparison data for four FPGAs.

3. Architectural Influences on the Mapping Process

Several features of Triptych's architecture -- the dual routing-and-logic nature of the RLBs, the highly constrained diagonal connections, and the disparity in delay characteristics between the diagonal and long line connections -- induce particular requirements for automatic logic synthesis, covering, placement, and routing.

One way to easily increase the RLB routing resources is to factor the logic into two-input rather than three-input logic nodes, thus assuring every RLB is open to at least one routing-only input. This produces a tradeoff between ease of routing and total RLBs used, since the number of logic nodes required increases when going from three-input to two-input factorization. Although this increase may be by as much as a factor of 4, in practice we generally encounter factors between 1.5 and 2. When routing placed circuits, additional routing resources can be obtained and congestion relieved by doing local re-synthesis. In addition, the different delay characteristics of the diagonal and long lines can be exploited when implementing high performance circuits – signals routed on vertical channels should be resythesized to have greater slack than signals on diagonals, because vertical channels have a greater delay.

Since placement algorithms generally make decisions based on an estimate of total routing length, it is important to realize that the short, diagonal connections induce in Triptych a non-standard distance metric, which is illustrated in Figure 5. Neither the Euclidean nor Manhattan metrics for this arrangement characterize Triptych's routing distance. For example, the cost to route one diagonal cell ahead is the same as to route one cell above. In addition, the distance a signal must travel from point a to point b may not necessarily be the same as from b to a.

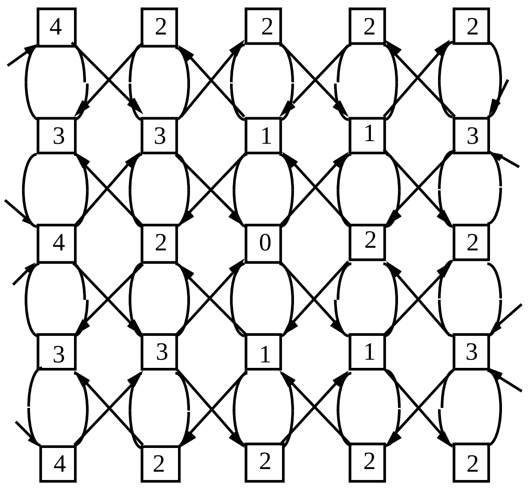

Figure 5. Triptych's distance metric. Distances are from the center RLB to each of the other RLBs.

Triptych's use of RLBs for routing as well as logic means that it is as important to sparsely place nodes in routing-congested areas as it is to find placements for each logic node. Clearly, the demand for routing resources at a particular location can only be determined by maintaining during placement an estimate of the expected routing congestion -- neither total interconnection length nor critical path length measures can provide this information.

While Triptych's short, diagonal connections allow for high-speed composition of RLBs, the tradeoff is reduced routing flexibility. This requires that local routing constraints be considered during placement. A given diagonal input to an RLB can come from only one of two adjacent RLBs. Since nearly every function will use at least one diagonal input, the signals available on the diagonals must be considered when we place a node in an RLB. One of the following three cases must occur in a valid placement:

1) the desired input is the function output of one of the diagonal RLBs
2) the desired input is a function input of one of the diagonal RLBs
3) one of the diagonal RLBs must have an unassigned input.

These conditions are necessary, but not sufficient to ensure routeability of the resulting placement.

4. Routing-directed Placement

Our automatic placement tool is built upon force-directed placement, but the modifications and extensions we make to this technique could be implemented within any iterative improvement placement method. Our tools try to create routeable placements by maintaining and meeting, for each cell we place, a limited set of local routing criteria which attempt to enforce some of the overall routing constraints. An RLB is considered to be locally routed if we have a placement for it such that:

1) we can assign signals to the cell's three incoming routing resources (two diagonals and one vertical wire) that provide the RLB's function inputs;
2) we can dedicate one of the RLB's outgoing lines or diagonals to route the signal calculated in the RLB;
3) any other signals which we route out of the RLB must be an input to the RLB's function, or else for each such distinct signal the RLB must have an unused input.

Note that the last condition favors adjacent placement of cells sharing inputs over adjacent placement of cells which do not share inputs. When we place an RLB, we insist that it meet the above routing constraints and that the cells around it continue to meet those criteria. If these conditions cannot be met for the target cell, then the closest routing-feasible location is chosen. In satisfying these conditions, we do allow the immediately surrounding cells to undergo some re-routing. In our current implementation, when evaluating a possible location for a single RLB, we examine the contents of as many as eleven cells, and may re-route as many as eight. Because Triptych's routing is tightly constrained, it is relatively easy to determine which other cells an RLB strongly affects; applying a similar treatment to an FPGA such as Xilinx, which has many more routing resources adjacent to each CLB, would require much more computation.

While the force-directed placement method tends to draw the placed cells closer together, the routing constraints tend to push them apart. In our scheme, if sufficiently many routing-only RLBs are interspersed among the RLBs implementing functions, the above conditions can be trivially met, and the result is a large, sparse placement. Clearly, the above criteria do not guarantee that the resulting placements will be routeable. However, any routeable placement must meet these restrictions. In particular, the more three input RLB functions there are, the less likely it is that the routing can be completed. If a cell and the source of its input are not adjacent, and the area between them is highly populated with three input logic nodes, the resulting congestion dramatically reduces routing options for the input.

To accommodate Triptych's distance measure, we have implemented the following twist on force-directed placement: we transform the regular two-dimensional (x and y) array of RLBs into a three-dimensional array, where the z-dimension separates the "red" and "black" cells in the "checkerboard" of the array; stretch the x and y coordinates of the cells to force local three-dimensional Euclidean distances to match the inherent routing distance; and then implement force-directed placement in three dimensions [Shahookar91]. We have found that inflating the z-dimension to nearly twice what it should be, thereby encouraging signals to flow in one direction, leads to placements which are generally easier to route and more closely resemble circuits which have been placed and routed by hand.

Currently we are implementing a routing tool based on Nair's simple global router [Nair87]. During an iteration of this procedure, each signal is ripped up and re-routed using a standard algorithm for shortest paths with the cost of using a given routing resource proportional to the number of signals using it. Initially multiple signals are allowed to be routed over the same piece of interconnect. However, after each iteration, all multiply used routing resources' costs are increased, with more highly congested resources receiving greater increases. In the next iteration, this encourages signal routes to avoid previously congested areas. Iterations continue until the circuit is routed or nearly routed.

At the current stage of our implementation, we must test our results by hand-routing the resulting placements. Table 2 gives the number of Triptych cells needed to place and route three random logic circuits. The first two circuits are the two smallest ISCAS benchmarks and the third is the well known traffic light controller. The figures given are

(1) The number of Triptych cells used after routing our algorithm's output. We used mostly 3-input logic functions during covering.

(2) The number of Triptych cells used for placement and routing in the hand-mapped version.

(3) A normalized figure for the number of CLBs used by MIS-FPGA to map the circuits to Xilinx

(4) A normalized figure for the number of CLBs used by xnfopt to map the circuits to Xilinx.

Note that while the Triptych figures have been completely routed, the Xilinx numbers are only a lower bound because they do not consider routeability. We scaled one Xilinx cell to 9.1 Triptych cells.

Circuit	Triptych auto-placed	Triptych hand mapped	Xilinx MIS	Xilinx xnfopt
s27	12	11	46	36
s208	130	59	173	127
tlc	46	24	73	73

Table 2. Results of mapping three sequential circuits to the Triptych (both automatically and by hand) and Xilinx architectures.

We expect that with planned improvements to our automatic placement and routing algorithms, our numbers will decrease to be closer to the hand-mapped results. Further work includes examining the trade-offs of considering larger and smaller surrounding areas when testing the placement of a single RLB for routeability; attempting to find the best local routing assignment for each cell placement, instead of simply finding one that works; determining the smallest ratio of two-input to three-input RLB functions that results in generally routeable placements; and occasionally performing more complete routes during placement in order to determine congested areas.

5. Conclusions

Current FPGAs generally fall into two categories: those with highly flexible interconnect, such as Xilinx, and those with extremely sparse, mostly local, and less flexible interconnect, such as CAL and CFA. The flexibly interconnected FPGAs are reasonably easy to route and can implement a wide range of functions, but their interconnect resources are significantly slower and occupy a much larger percentage of the chip area than more limited interconnects. Inflexible interconnects are fast and small, but severely limit the types of circuits these FPGAs can be used to implement. We feel the solution to this problem is to develop and use FPGAs whose routing resources fall in the middle ground. We have introduced such an architecture, Triptych, and demonstrated that using an unrefined placement method which considers local routing conditions yields circuit mappings that are comparable to those achieved on Xilinx.

Acknowledgements

Thanks to Chris Hébert, David Song, and the other students of CSE568, Winter quarter 1991 for contributing to the early design phases of the Triptych layout and mapping techniques. Thanks also to Larry McMurchie and Sung-Eun Choi. This research was funded in part by the Defense Advanced Research Projects Agency under Contract N00014-J-91-4041. Gaetano Borriello and Carl Ebeling were supported in part by NSF Presidential Young Investigator Awards, with matching funds provided by IBM Corporation and Sun Microsystems. Elizabeth A. Walkup was supported by a National Science Foundation Graduate Fellowship.

References

[Actel91] Actel Corporation, “ACT Family Field Programmable Gate Array Data Book”, 1991.

[Algotronix91] Algotronix Limited, “CAL1024 Preliminary Datasheet”, 1991.

[Beetem91] J. F. Beetem, “Simultaneous Placement and Routing of the Labyrinth Reconfigurable Logic Array”, International Workshop on Field-Programmable Logic and Applications, Oxford, 1991.

[Carter86] W. Carter et al., “A User Programmable Reconfigurable Gate Array”, Proceedings of the IEEE Custom Integrated Circuits Conference, May 1986.

[Ebeling91] C. Ebeling, et. al., “Triptych: A New FPGA Architecture”, International Workshop of Field Programmable Logic and Applications, Oxford, September 1991; also appears at University of Washington, Department of Computer Science & Engineering, Technical Report 91-09-05.

[Hauck92] S. Hauck, et. al., “Triptych: An FPGA Architecture with Integrated Logic and Routing”, Advanced Research in VLSI and Parallel Systems, to appear March 1992;

[Kean89] T. Kean, “Configurable Logic: A Dynamically Programmable Cellular Logic Architecture and its VLSI Implementation“, PhD Thesis, Dept. of Computer Science, University of Edinburgh, 1989.

[Nair87] R. Nair, "A Simple Yet Effective Technique for Global Wiring", IEEE Transactions on CAD, vol. CAD-6 no. 2, 1987.

[Schlag91] M. Schlag, P.K. Chan, and J. Kong, "Empirical Evaulation of Multilevel Logic Minimization Tools for a Field Programmable Gate Array Technology", International Workshop on Field-Programmable Logic and Applications, Oxford, 1991.

[Shahookar91] K. Shahookar, P. Masunder, “VLSI Cell Placement Techniques”, ACM Computing Surveys, Vol. 23, No. 2, June 1991.

[Xilinx91] Xilinx, Inc., "The Programmable Gate Array Data Book", 1991.