Effects of FPGA Architecture on FPGA Routing

Stephen Trimberger
Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124 USA
steve@xilinx.com

Abstract

Although many traditional Mask Programmed Gate Array (MPGA) algorithms can be applied to FPGA routing, FPGA architectures impose critical constraints and provide alternative views of the routing problem that allow innovative new algorithms to be applied. This paper describes routing models provided by some commercial FPGA architectures, and points out the effects of these architectures on routing algorithms. Implicit in the discussion is a commentary on current and future research in FPGA routing.

1 Introduction

Fundamentally, field programmability gets in the way of routing. An individual MPGA route consists of a low-resistance, low-capacitance metal path (Figure 1a). In contrast, FPGA interconnect consists of a network of interconnect segments, loaded with switches and connected with high-resistance switches. SRAM programming cells control pass transistors that steer signals to make routing paths (Figure 1b). Antifuses connect wiring segments when programmed (Figure 1c). Although different FPGA technologies have different resistance and capacitance numbers, and although FPGA makers can often control the actual values of these parameters with clever circuit design, all the numbers for FPGAs are larger than the corresponding values for MPGAs.

Given the large resistances and capacitances involved, it is no surprise that FPGA routing delay is critical. FPGA interconnect delay may dominate logic block delay, but this is qualitatively no different than MPGA interconnect. It is not the magnitude of the delay, but the character of the delay that causes routing problems. FPGA delay comes in large increments, so a small change in path length can make a large difference in delay. Even short paths must be modeled as RC trees.

Further, the metal segments that make up FPGA routing are not interchangeable. Large circuitry is required to support field programming, so FPGA manufacturers restrict the connectivity of interconnect to save area. Therefore, the wiring segments typically are built in a variety of lengths. They may connect to a subset of adjacent segments and to a subset of logic block pins. The partial-connectivity pattern may be different for different segments in the same channel. A router that does not distinguish among the different types of interconnect will fail to produce a high-quality result.

Because FPGA interconnect is more delay-intensive then MPGA interconnect, FPGA logic blocks tend to be larger functions, reducing the need for interconnect. The large size of FPGA logic blocks coupled with the generality needed to field-program the blocks yields logic blocks with large numbers of logically-equivalent pins (or pins that can be made logically equivalent by simple reprogramming of the logic block). To successfully route an FPGA, routers must exploit these logically-equivalent pins directly or by re-programming the logic blocks.

On the positive side, FPGAs provide routing software with a logical connectivity model that may be less constraining than the multi-level 2-dimensional model of the physical layout. For example, logic block pin connections can be chosen to avoid vertical channel constraints. The wire segments in an FPGA are logical connections, and do not necessarily reflect the layout. The fixed interconnect segments may be made up of layout on several mask layers, so they can cross one another, giving the illusion of an arbitrary number of non-interfering wiring layers.

Furthermore, many nasty routing problems are effectively addressed by FPGA architectures. Most FPGAs include dedicated clock routing, so no clock router is needed. Long-distance buffered segments may be provided for high-fanout

32nd ACM/IEEE Design Automation Conference ®

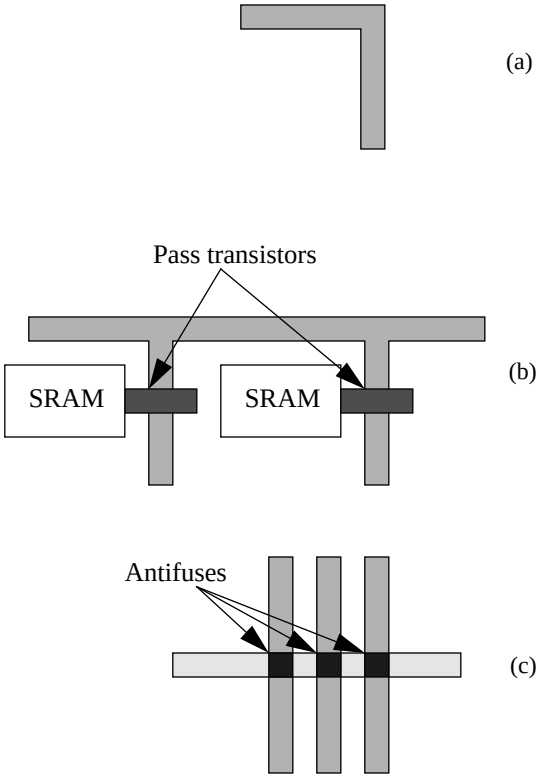

Figure 1. Field programmability loads wiring paths. a) MPGA metal trace. b) SRAM with pass transistor. c) Antifuses.

signals, so the placer may not need to deal with buffer trees, though the router must build low-skew signal distribution paths using these resources. At the same time, other features may cause new routing problems. For example, logic blocks may have very restricted internal connections which may be fine when building logic, but are too restricted for general-purpose detailed routers. Such paths may be effectively routed with a pattern-directed router, rather than by propagating a maze-route wavefront.

The following two sections detail several features of symmetrical FPGAs and segmented channel FPGAs, and their implications for routing software. Although examples are taken from commercial FPGA architectures, it is not the intent here to survey all FPGA architectures.

2 Symmetrical FPGAs

A symmetrical FPGA [Brown 1992] consists of an array of logic blocks. The wiring channel is made up of wire segments in a variety of fixed lengths. Logic block and I/O block pins connect to wire segments in wiring channels, and wiring channels intersect at switchboxes (Figure 2). Symmetrical architectures are common with SRAM-programmed FPGAs. SRAM cells control pass transistors for interconnect, as shown in Figure 1b.

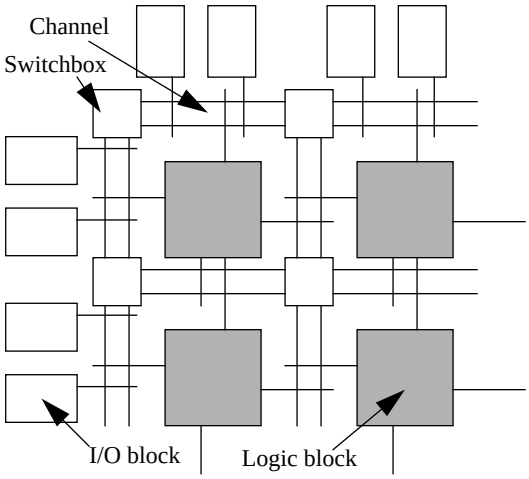

Figure 2. Symmetrical FPGA architecture.

Ideally, there will be a switch at the intersection of every two segments in the switchbox and at the intersection of every wire segment in the channel with every logic block pin. The large size and the capacitive load of the pass transistors make this impractical. Instead, relatively few of the connections are possible. As shown in Figure 3a, not every track in the channel is accessible to every pin on the logic block. Routing a signal to a channel adjacent to the logic block may not be sufficient to complete the net. Logically-equivalent pins on a logic block may appear in different channels, as shown in Figure 2. Further, the switchbox connectivity may be limited, e.g., in the Xilinx XC4000 switchbox, each of the 8 single-length routing segments is allowed only three of the 31 possible connections in the switchbox (Figure 3b).

The constraints imposed by this limited connectivity tightly couple global and detailed routing. The limited connectivity in the channel couples the pin assignment to the track assignment for the net. The switchbox constraint couples the track assignment for a net in the detailed routing of a channel to the track assignment in other channels. Fundamentally, the symmetrical FPGA routing problem cannot be decomposed spatially, as is typically done with gate arrays, where a global router divides detailed routing into several independent problems. A router for this kind of FPGA must perform channel selection simultaneously with track assignment.

On the other hand, in addition to the constraints, the FPGA architecture may be more flexible than the simple two-dimensional model familiar to MPGA developers. For example, in Figure 3b, any of the six connections among the four lines in the switch box is allowed independently of the others. The switch box allows two signals to cross over one another without interference. It also allows knock-knee routing, where for example, segments A and B can be connected in one net, while C and D form another.

Finally, since all resources on the FPGA are predefined and fixed, unused logic resources may be used by the FPGA router as routing resources. This allows the router to buffer signals, and to add routing paths through logic blocks. Although this type of routing path may be considered a target of opportunity in most routers, cellular architecture FPGAs, such as the Algotronix CAL and the Atmel AT6000 series, rely on such routing through logic cells as their primary interconnect mechanism. Commercial routers re-assign pins, buffer signals and even duplicate logic to complete routing paths. While essential for commercial success, these features do not appear in the literature.

One commercial router reported in the literature [Frankle 1992] is a single-phase maze router with ripup and re-route. There is no global route phase. The router is path delay driven, using a slack allocation and relaxation method. It uses net delays to drive wavefront expansion, pruning the search space when path delays are too large. Delays are calculated by an incremental Penfield [1981] model, using parameters derived from measurements from silicon characterization. The router may use buffers provided in the wiring channels in the architecture. Buffers are modeled in the delay calculation as a combination of fixed delay, capacitance load and drive resistance. The router may route signals through unused logic in logic blocks, configuring that logic as buffers. In addition, the router may run incrementally [Trimberger 1994], allowing a user to place and route part of a design, then incrementally add to the design. This capability exploits the reconfigurability of the FPGA. The design may be developed and tested incrementally, with in-system testing of increasingly larger pieces until the design is complete. Incremental routing is also used to insert “probe points”, user-generated observation points to facilitate in-circuit debugging. These probe points are removed in the final design.

The sparse connectivity of FPGA interconnect can also allow a fundamentally different approach to routing. The CGA router and its descendant SEGA [Brown 1992] generate a set of possible routing paths for every net. Then, the router iteratively selects paths for nets that are forced to take a specific path or that preserve the most options for other nets. The nets are routed in order of most-constrained to least-constrained,

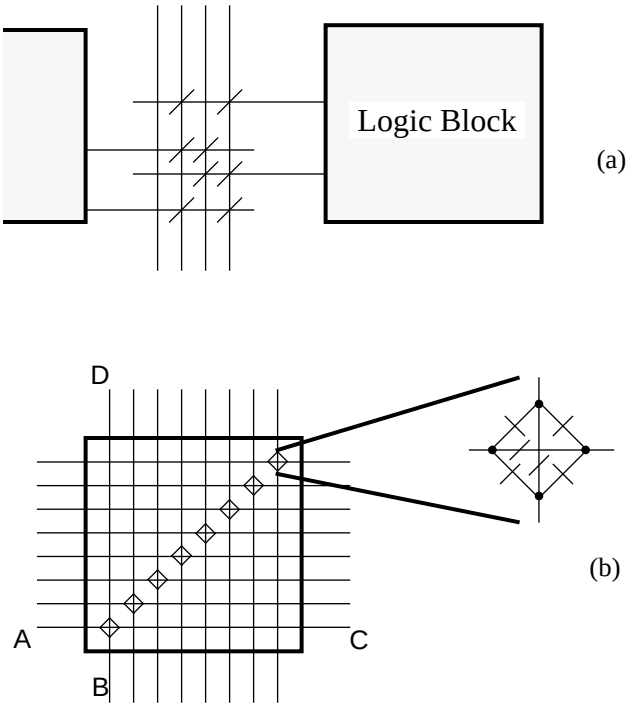

Figure 3. Limited connections in FPGA interconnect.
a) Logic block to channel. b) Switchbox.

with paths chosen to minimize conflict with still-unrouted nets. In a gate array, this kind of detailed router is impractical because of the huge number of potential paths, but in an FPGA with limited connectivity, the number is manageable and the algorithm is able to simultaneously consider enough different paths to produce a good result. The CGE algorithm can be extended to target timing by annotating net delays on the route path sets. Paths with excessive delays can be eliminated from consideration. This would seem to be an obvious improvement to this work, but again it has not been reported in the literature.

3 Segmented Channel FPGAs

An antifuse is a via-sized one-time-programmable device which conducts after a high current is passed through it. Antifuses are small and have low capacitance, so they are used liberally. An FPGA may have an antifuse at the intersection of every horizontal and vertical wiring segment. Despite their small size, antifuses have significant series resistance, so wiring channels are typically composed of wire segments in a variety of sizes: a segmented channel FPGA architecture.

Figure 4 shows part of a simple segmented channel architecture modeled after the Actel ACT2 FPGA. The channel includes an antifuse at every intersection of a horizontal and vertical wiring segment. In addition, the circles denote antifuses that may be used to connect two wiring segments end-to-end in a track to make a long route. As with the symmetrical model described earlier, this is a logical view; physically, the routing channel may be laid on top of the logic.

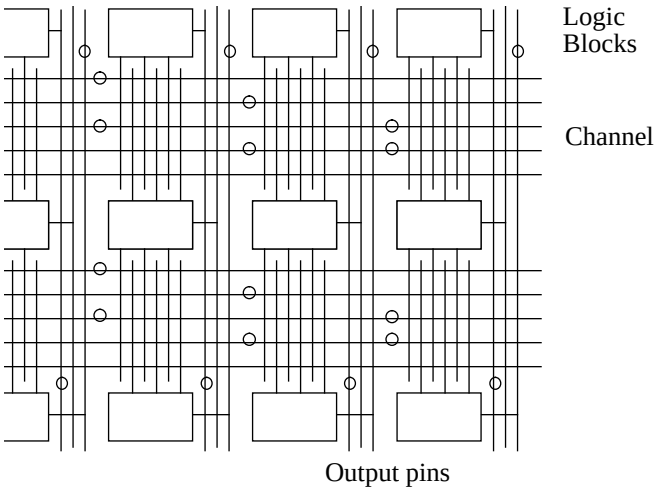

Figure 4. Segmented channel FPGA architecture.

The distribution of lengths of the tracks in the channel is fixed for the FPGA family. For full flexibility, one would like the channels to be made up of short segments, each perhaps a single logic block pitch. Although the antifuse capacitance and on-resistance are low, a large number of antifuses connected in series would make nets intolerably slow. Further, in some commercial architectures, the segmentation is constrained by the way the antifuse programming is done, so these architectures permit no more than a fixed number of segments on a track. As a result, tracks consist of segments in a variety of lengths, often in an irregular segmentation pattern.

The small size of the antifuse allows full or nearly-full population of intersection points with antifuses. Therefore, in antifuse architectures, routing can be decomposed spatially with a global router that selects paths through routing channels, then a detailed router that performs track assignment independently for each channel. In Figure 4, the logic block outputs are shown driving vertical segments that span three channels. In this case, the placement may be constrained to place all loads on a net in the rows that are reachable by the output segment of the logic block that sources the net. The most efficient router in this kind of constrained problem may be a pattern-driven router, which looks for optimal connection patterns given a limited set of possible routes for a net.

As with the symmetrical architecture, segmented channel architectures can be built with connections into the channel that avoid all vertical channel constraints. This simplifies the channel routing problem (and is more efficient). Traditional channel routing algorithms, such as those based on bin packing, can be applied to this problem. However, the issue of timing-driven routing remains. Many approaches to performance-driven segmented channel routing address the problem of limited-segment routing [Greene 1990]. In 1-segment and 2-segment routing, the router attempts to do a track assignment that limits the number of serial antifuses in the path of each net to zero or one, in order to limit the delays incurred by the serial resistance of the antifuses. This limitation increases the complexity of the problem [El Gamal 1991]. Fundamentally, though, while these limited-segment routing algorithms are intended to limit delay, they do so only indirectly. Path-delay optimization is required for routing large segmented channel FPGAs, just as it is for symmetrical FPGAs. Although such techniques exist in commercial systems, they seem to be ignored in the research community.

4 Futures and Conclusions

FPGAs include three kinds of resources: I/O, logic and routing. Of these three types of resources, I/O capacity and logic utilization can be measured easily. However, routing utilization is difficult to measure, and therefore it is usually difficult to estimate when the FPGA capacity limit is reached due to interconnect usage. Since interconnect dominates FPGA area, and since signal delay depends on the capacitance due to potential connections, additional FPGA interconnect is very expensive. FPGA vendors are understandably reluctant to add generous amounts of extra wiring resources to remove this concern. As a result, routing and routability prediction will likely remain important topics in FPGA routing research.

FPGAs are typically built by defining a tile that contains both the logic and the interconnect, then building an array of those tiles. As process technology improves, FPGA vendors are able to build larger arrays of these identical tiles. As they do, routability degrades because proportionally more interconnect is required on large devices [Heller 1978][ElGamal 1981]. Therefore, an FPGA family will necessarily contain small devices that are easy to route and large devices that are difficult to route. Routing software will always be graded on its performance on the difficult problems.

FPGA routers are being required to routinely perform tasks that are not typical in MPGA routing. As FPGAs grow in complexity, FPGA routers increasingly represent the cutting edge of router research. True path-delay-driven algorithms

have been part of FPGA routers for years. Incremental routing, signal buffering and logic duplication are state of the art in FPGA routers.

Finally, while this paper has discussed the effect of FPGA architecture on FPGA routing, there is also the issue of the effect of routing technology on FPGA architecture. Increased expertise and research in routing leads to a better understanding of the constraints on FPGA architectures. This understanding will lead to FPGA architectures that work more effectively with routing software. As architectures change, we can expect new problems to be presented to routing software, but hopefully those problems will also be somewhat easier to solve.

5 References

S. Brown, et. al., *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, 1992.

A. El Gamal, J. Greene, V. Roychowdhury. "Segmented Channel Routing is Nearly as Efficient as Channel Routing (and Just as Hard)," *Proceedings. VLSI Conference*, Santa Cruz, CA, March 1991.

A. El Gamal, "Two-Dimensional Stochastic Model for Interconnection in Master Slice Integrated Circuits," *IEEE Transactions on Circuits and Systems*, vol. CAS-28, no. 2, February 1981.

J. Frankle, "Iterative and Adaptive Slack Allocation for Performance-driven Layout and FPGA Routing", *Proceedings of the 29th Design Automation Conference*, 1992.

J. Greene, V. Roychowdhury, S. Kaptanoglu, A. El Gamal, "Segmented Channel Routing," *Proceedings of the 27th Design Automation Conference*, 1990.

W.R. Heller, et. al., "Prediction of Wiring Space Requirements for LSI," *Journal of Design Automation and Fault Tolerant Computing*, May 1978.

P. Penfield, Jr., J. Rubenstein, "Signal Delay in RC Tree Networks," *Proceedings of the 18th Design Automation Conference*, 1981.

S. Trimberger, ed., *Field-Programmable Gate Array Technology*, Kluwer Academic Publishers, 1994.

Y. Wu and M. Marek-Sadowska, "Graph Based Analysis of FPGA Routing", *Proceedings of Euro-DAC*, IEEE, 1993.