A Novel Predictable Segmented FPGA Routing Architecture

Emil S. Ochotta, Patrick J. Crotty, Charles R. Erickson, Chih-Tsung Huang,
Rajeev Jayaraman, Richard C. Li, Joseph D. Linoff, Luan Ngo, Hy V. Nguyen,
Kerry M. Pierce, Douglas P. Wieland, Jennifer Zhuang, and Scott S. Nance

Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124 USA
408-559-7778
emil@xilinx.com

1. ABSTRACT

In the development of new FPGA architectures, a designer must balance speed, density and routing flexibility. In this paper, we discuss a new FPGA architecture based on a patented [1], novel, segmented routing fabric that is targeted to high performance and predictability but does not sacrifice routability or area efficiency. Current segmented architectures allow much flexibility in routing, but incur large delay penalties when a signal has high fanout or must traverse medium to long distances to reach its target. Reducing the number of programmable interconnect points (PIPs) that a signal must traverse to reach its target, while eliminating the RC delay buildup due to signal fanout, improves design performance and offers highly predictable signal delays.

1.1 Keywords

FPGA, Programmable Logic, Routing.

2. INTRODUCTION

Since the introduction of the FPGA in 1985 [2], the number and scope of applications in which FPGAs are used have increased dramatically. One reason for this dramatic growth has been the shrinking gap between FPGAs and application specific integrated circuits (ASICs) in terms of capacity, cost, usability, and performance. It follows that the goal of architects creating a new FPGA is to reduce and eventually eliminate this dwindling gap. This paper examines one aspect of FPGA architectural design, design of the routing fabric, and presents a patented [1], novel, segmented routing architecture, that has significant advantages in performance and usability while balancing the demands of cost and capacity.

We organize the balance of this paper as follows: In the next section, we describe the goals and key ideas behind our new routing architecture. We then set the context for the routing architecture by providing an overview of the FPGA of which it is a part and a brief description of how the routing architecture evolved. This is followed by the core of the paper, which describes the routing fabric and presents some results from our analysis of it. Finally we close with some conclusions.

3. GOALS AND BACKGROUND

Our new FPGA routing architecture is designed to maximize performance and predictability, while maintaining high routability and efficient area utilization. These new goals — performance, predictability, routability, and area efficiency — result from applying the more general goals of capacity, cost, usability, and performance to the FPGA routing design problem. For some of these goals, the effect is readily apparent: routing performance, the delay of the interconnect from one logic element to the next, directly influences the overall performance of a user's design in an FPGA. In FPGAs, routing delays have always been important in computing overall performance, and as process minimum feature sizes continue to shrink, routing delays increasingly dominate logic delays. Similarly, the effect of area efficiency on cost is direct, since the cost of an FPGA is proportional to its die area. For predictability, the relationship to the more general goals is not so obvious. Predictability is the ease and accuracy with which interconnect delay can be estimated for a design when the gates have been placed in the logic elements on the FPGA but the routing has not been completed. Good predictability makes it easier to write software that can quickly implement a user's design on the FPGA, significantly impacting its usability. Routability also influences usability. Routability is a measure of how easy it is to interconnect the necessary logic elements to complete the implementation of the user's design on the FPGA. Again, greater routability makes it easier to write efficient routing software, thereby increasing usability. Routability also impacts cost and capacity. If there are insufficient routing resources to interconnect all the placed logic elements, routability can be the limiting factor in the capacity of a device. Similarly, since cost is directly proportional to die area, for a given amount of logic that can be interconnected, a routing architecture that consumes more area is more expensive.

Now that we have established the relationship between our routing specific goals — performance, predictability, routability, and area efficiency — and our more general goals for FPGA architecture design, it is important to identify specific areas for improvement in existing FPGA routing architectures. One of the most important of these is the build-up of RC signal delay through unbuffered segments of the programmable interconnect. This delay can have a dramatic negative effect on performance, particularly for high-fanout nets, where each load contributes additional delay. Signal delay also can be exacerbated by poor routability because a connection may be relegated to a suboptimal path due to signal congestion in the local area. The non-linear nature of RC trees also makes the delays on routes with many unbuffered segments difficult to predict. When the problem is further complicated by the possibility that a signal may be forced to route around congestion, predictability becomes nearly impossible. Because of these weaknesses, a key idea behind our work is to eliminate unbuffered segments from the routing architecture.

Another key idea in our routing architecture comes from trying to capture the best features of interconnect structures that are too costly to build. The ideal FPGA routing architecture for performance, predictability, and routability is a fully populated crossbar switch, which allows any logic element to connect to any other logic element on the FPGA through a minimum number of programmable interconnection points (PIPs), the programmable connections between wires. Unfortunately, the amount of wiring needed for a crossbar grows quadratically with the number of logic elements, so a full crossbar switch is not area efficient. For an FPGA with a capacity of more than a few thousand gates, a crossbar is too expensive; however, we have captured some of the flavor of a crossbar in our routing architecture without paying the high area cost.

Our new routing architecture combines the key idea of buffering segments with the flavor of a crossbar structure. In a somewhat whimsical allusion to the Gordian knot, the architecture is called “Alexander”, a name we shall use for convenience in the balance of this paper. Our Gordian knot was the problem of achieving our performance and predictability objectives while maintaining routability and area efficiency. However, Alexander cut through his knot problem and so shall we. We begin the description of our solution with the most general view of the Alexander routing architecture and the FPGA of which it was a part, then motivate some of the features in the architecture by outlining its evolution, and finally describe the architecture in detail.

4. FPGA OVERVIEW

Before we describe the Alexander routing architecture in detail, we first provide a context for the routing architecture by describing the rest of the FPGA of which it was a part. The general arrangement of the Alexander FPGA is shown in Fig. 1 and is similar to the coarse grain static RAM architecture of the Xilinx XC4000 Family [3]. The architecture consists of a two-dimensional array of logic elements called Configurable Logic Blocks (CLBs), that are interconnected by the Alexander routing. A single CLB and its associated routing resources are collectively referred to as a tile. As shown in Fig. 1, the CLB tiles form the core of the FPGA and are surrounded by a ring of programmable I/O buffers. The CLBs implement the user’s logic, and the I/O buffers provide the interface between the FPGA core and the external world.

As shown in Fig. 2, each CLB consists of two logic cells. These logic cells are largely independent: they have separate data inputs and outputs but share the control signals on the flip flops. As shown in Fig. 2, each logic cell consists of a function generator that can be configured to produce any function of its four inputs, and an edge-triggered D flip flop that can act as a storage element.

Fig. 3 presents a somewhat more detailed view of a logic cell. Combinational logic is implemented in the function generator, and the flip flop can store a single bit of state information. The function generator and flip flop in a logic cell are arranged in series such that the output of the function generator can be used as the input to the flip flop. To provide fast and compact arithmetic, the logic cell

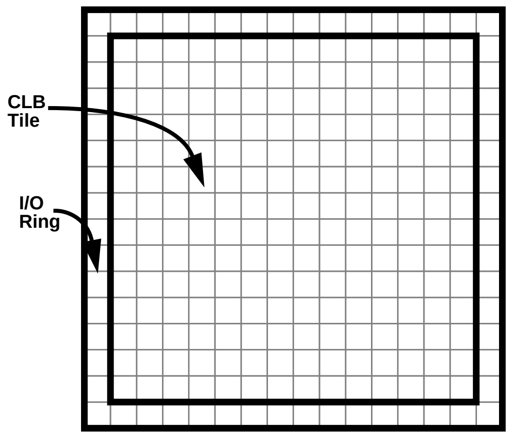

Figure 1. Architecture Layout.

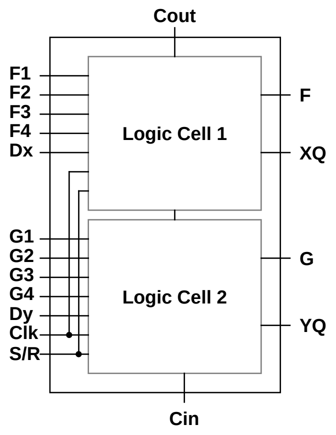

Figure 2. Configurable Logic Block (CLB)

implements full adder circuitry and a fast carry chain. The carry circuitry uses the function generator inputs as the operands and a carry-in signal propagated from the logic cell immediately below it, generating carry-out for the logic cell above it. The Dx pin on the logic cell provides a direct path to the flip flop, allowing it to be used independently of the function generator. As we describe in Section 6, the simple but powerful combination of features in the logic cell are a good match for the Alexander routing architecture.

For those intimately familiar with other FPGAs, it is likely apparent that we are neglecting details of the CLB in Fig. 2 and Fig. 3. In fact, there are several features of the Alexander FPGA that we do not detail because they are only indirectly related to our novel routing architecture, yet some of these features are worth mentioning for completeness. First, there are several global clocks for distribution of clock signals and other very high fanout nets. Second, there are fast dedicated direct connects from each logic cell to its neighbor on the right, to allow construction of functions wider than the four-inputs allowed by the function generators. Finally, there are long-lines that provide additional routing resources throughout the FPGA. We exclude all these resources from further discussion.

5. GENESIS OF THE ROUTING FABRIC

Before we present the details of the Alexander routing architecture, we first motivate our discussion with a simplified history of its evolution. The goal of the Alexander routing architecture was to improve on the routability, performance, and predictability of the Xilinx XC4000 architecture, and for this we were willing to pay a modest area overhead. As a starting point for this new architecture, we began with a crossbar interconnection scheme as shown in Fig. 4. This conceptual diagram depicts CLBs with only a single input and output. The greatest advantage of this scheme is the performance benefit of being able to connect any two CLBs with only a single PIP. In Fig. 4, connecting the CLB in the upper left hand corner to the CLB in the lower right hand corner requires enabling only the circled PIP. However, there are significant drawbacks that make a full crossbar interconnection scheme impractical for large, commercial FPGAs. The first of these is the amount of chip area consumed by the routing. A chip-length wire is required for each input or output from each CLB, and a PIP is required between every input and output

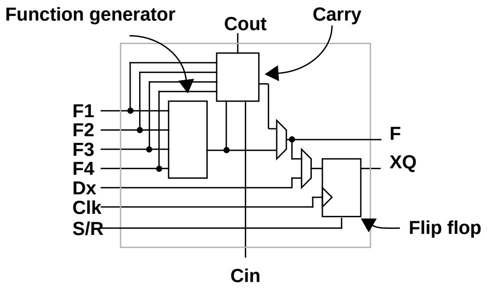

Figure 3. Logic Cell

wire. With hundreds or thousands of CLBs, the three-by-three arrays of PIPs in Fig. 4 would quickly grow until they were economically impractical. Another important result of a full crossbar is that wires that run from one edge of the FPGA to the other are required for each input and output. These wires would be slow for large devices simply because of their length and the large electrical load due to the many PIPs on them, as can be clearly seen in other non-segmented routing architectures, negating the advantage of the single-PIP connections. Finally, a more practical concern of the crossbar is that it cannot be tiled, which means that creating a ten percent larger device of the same family would require a complete re-layout, which is time-consuming and expensive. The genesis of the Alexander architecture was to try to overcome these area and tileability deficiencies while preserving the primary advantage of the crossbar — the ability to interconnect any two CLBs with a single PIP.

The first step towards the Alexander architecture was to simply cut the input and output wires in Fig. 4 so that instead of running the length of the chip they covered only two tiles. The result is the routing scheme in Fig. 5. This routing scheme preserves single-PIP connectivity to nearby CLBs but is tileable (ignoring the edge effects visible in Fig. 5). In addition to edge effects, the simplified routing fabric in Fig. 5 has a directionality bias because each CLB has a single input and output. The first step towards making this idea practical was to increase the number of inputs and outputs, balancing them so that they extend in all directions from the CLB. As we shall see when we describe the routing architecture in detail, this characteristic pattern (visible in Fig. 6) is still a part of the routing fabric.

From this simple beginning, the routing architecture evolved through five major changes and several minor revisions. This evolutionary process was guided primarily by routing experiments performed with Xilinx' PPR software that was modified to route each new version of the Alexander architecture. To measure the success of each new variant,

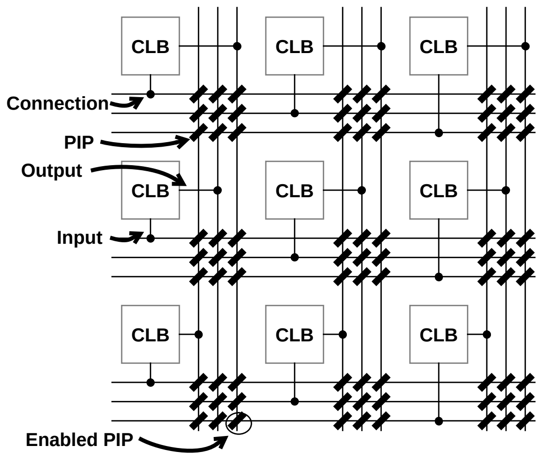

Figure 4. Crossbar interconnection scheme. Connecting the upper-right CLB to the lower-left CLB requires enabling only the circled PIP.

statistical comparisons of routability were performed using a suite of 56 designs that were placed and routed. These designs were selected both because they were difficult to route in XC4000 devices, and because they covered a broad range of customer design styles. After each modification of the routing architecture, designs that routed poorly were examined in detail and changes were proposed to address the specific weaknesses found. These changes were then analyzed to predict the impact on FPGA area and on performance. This evolutionary step was repeated until the architecture converged to the form we describe in the balance of this paper.

6. ROUTING FABRIC

Now that we have described the path by which the Alexander routing architecture evolved, we present a detailed description of the routing fabric itself. The Alexander routing fabric is created from three inter-woven elements: the extended input pin (EIP), the multiplexor complex (Mux-C), and the output complex wire (OCW). To understand the overall fabric, it is essential to first understand each of these threads in isolation and then understand their interconnection.

The first element in the routing fabric is the extended input pin (EIP). Recall from Fig. 2, that each CLB has twelve inputs pins. These pins are divided into four groups of three pins each. As shown in Fig. 6 each group of three input pins extends for a distance of two tiles towards a different edge of the FPGA. As can be seen by comparing Fig. 6 with Fig. 5, the EIPs have not changed significantly from the original concept for the architecture. However, the same cannot be said for the outputs from the CLBs.

Instead of a structure like the EIPs, the outputs from the CLB drive the second of the elements in the routing fabric: the Mux-C. The Mux-C is the intersection where signals can change direction or extend themselves on their way from CLB output to CLB input. Recall from Fig. 2, that each CLB has four outputs to the routing fabric. Although there is a single Mux-C and a single CLB in each tile, the arrows in Fig. 7 show that each of the CLB outputs drives a different one of the four Mux-Cs that surround it. Fig. 8 shows in greater detail how the CLB outputs connect to the surrounding Mux-Cs. Each function generator and flip flop in the CLB has two possible ways of connecting to the adjacent Mux-Cs. This is shown in Fig. 8 where the F function generator connects to two Mux-Cs: one in the upper-right and the other in the lower-left. Similarly, the G function generator, the XQ, and YQ flip flop outputs connect to two Mux-Cs as shown in the figure. Note the asymmetry between the connection of the two logic cells in a logic block. The fact that the two logic cells in a logic block are identical in all respects except their connectivity to

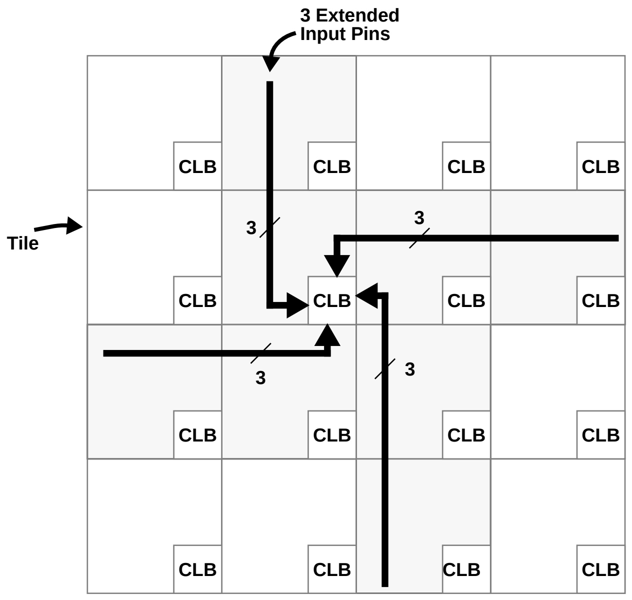

Figure 6. Extended input pins bring inputs directly into the CLB from a distance of two tiles away.

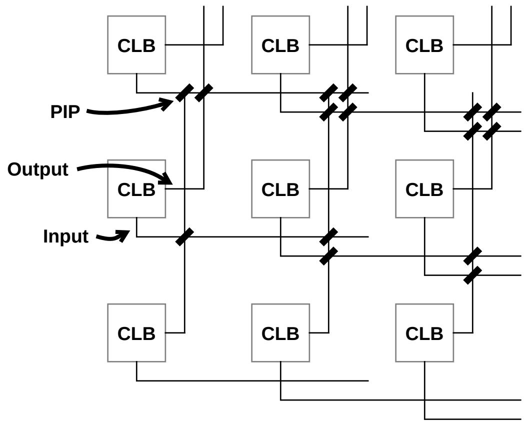

Figure 5. Modification of the crossbar that is the foundation of the Alexander routing architecture.

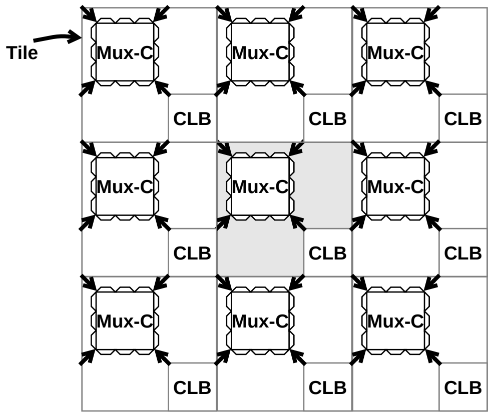

Figure 7. The multiplexor complex (Mux-C) collects outputs from each of the 4 surrounding CLBs.

the Mux-Cs provides the flexibility to fix routing problems with small placement changes. Because all CLB outputs go to a Mux-C, it acts as a collection point from which signals leaving the CLB are distributed to other routing resources.

The third element in the routing fabric is the output complex wire (OCW). A single OCW is shown in Fig. 9. As shown in Fig. 10, four OCWs begin at each Mux-C, each one of the four extending towards a different edge of the FPGA. Thus, a signal leaving a CLB output goes through a Mux-C and on to an OCW. The OCW is the element that ties the routing architecture together because it is driven by a Mux-C and connects to EIPs and other Mux-Cs. To describe how the connections to the EIPs are made, we must describe the component wires that make up the OCW. Referring to Fig. 9, an OCW is in fact composed of five different wires: two double-PIP (DP) and one each of double-metal (DM), quad-PIP (QP), and quad metal (QM). A wire is called a double or a quad depending on whether its length is two or four tiles respectively. Thus, a double-PIP wire is a wire that extends across two tiles and has several PIPs on it. In the Alexander routing architecture, the EIPs are connected by PIPs to the double-PIP and quad-PIP wires. As shown in Fig. 11, the double-PIP wires are connected with PIPs to every EIP that they cross. The quad-PIP wires are connected only to the EIPs they cross in the third and fourth tile they traverse. Wires with PIPs are flexible resources but slower than wires without PIPs because each PIP adds an electrical load to the wire. In contrast, the double-metal and quad-metal wires are express lines that have connection points only at their ends. In the context of the OCW, we can now show how the reach of the EIPs provides not only routability, but additional speed as well. The two-tile input reach on the pin provides a direct (*i.e.* fast) path directly into the CLB from a greater distance. Experimental results show that the extended input pins also create additional routing flexibility by providing convenient routing paths from the OCW to CLBs in any of the adjacent rows of tiles, as shown in Fig. 11**.** The combination of OCW and EIP is both flexible and efficient because it provides the PIP wires to maximize routability and the metal wires to maximize speed.

The Mux-C, OCW and EIP are interwoven in a segmented routing fabric that is difficult to draw because of its complexity. In this fabric, each tile contains a CLB driven by four EIPs and a Mux-C that drives four OCWs. Because

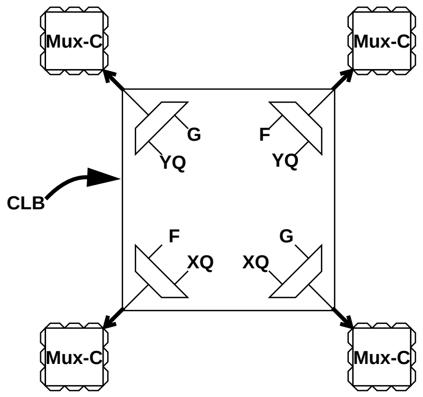

Figure 8. Connections to the routing fabric

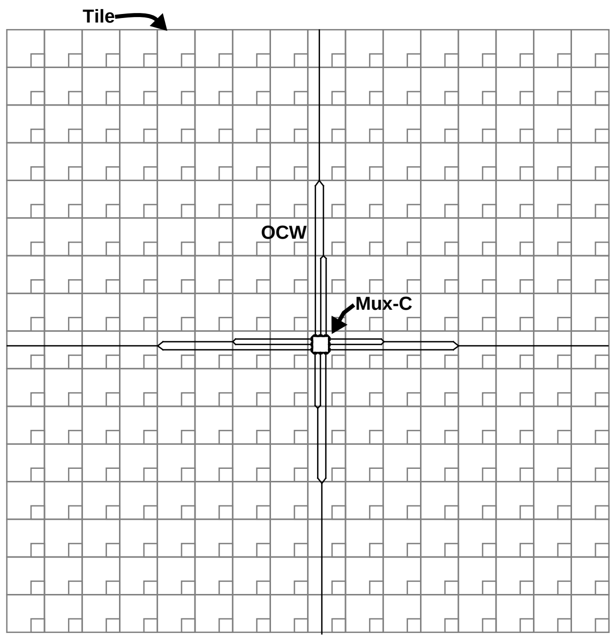

Figure 10. An OCW extends from each of the four sides of every Mux-C.

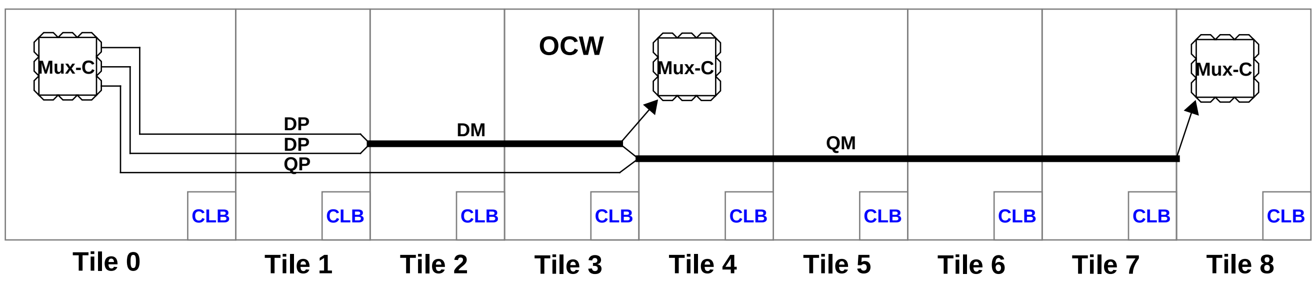

Figure 9. The output complex wire is composed of two double-PIP wires (DP) and one each of double-metal (DM), quad-PIP (DP), and quad metal (QM).

a given OCW has a reach of eight tiles, and OCWs proceed in all four directions, each tile actually contains wires from 32 different OCWs. Similarly, there are actually 24 EIPs in the tile and 288 PIPs to interconnect the OCWs and EIPs.

It is also worth pointing out that this combination of Mux-C, OCW, and EIP still retains some of the flavor of a crossbar switch because many CLB-to-CLB connections can be made with a single PIP. Although the connectivity is not complete, as we can see in Fig. 11, each CLB is densely connected to the CLBs nearby via a single PIP connection between the OCW and the EIPs. This nearest neighbor connectivity is what is needed to efficiently route the majority of signals in a typical design. As the distance from source to target increases, the amount of connectivity decreases, and the structure looks less like a crossbar. The drastic reduction in the number of PIPs reflects the need for an area-efficient routing architecture. To maintain good connectivity for long routes (typically signals with high fanout) the OCW provides the metal wires to quickly traverse the FPGA.

When a signal must travel a long distance, the OCW is fast because of the double-metal and quad-metal wires. Signals that extend beyond eight tiles use more than a single OCW by routing through additional Mux-Cs. To facilitate this, as shown in Fig. 9, the OCW is connected to the nearest Mux-C at the end of the OCW's double-metal wire and its quad-metal wire. From the Mux-C, the signal can take any of the four OCWs it drives, allowing the signal to change directions and get to any corner of the FPGA very quickly.

A more detailed view of a single edge of a Mux-C is shown in Fig. 12. Since there are three wires at the beginning of each OCW and four OCWs driven by each Mux-C, there are twelve multiplexors in each OCW, and each set of three OCW-driving multiplexors is identical. Each multiplexor is an 8-to-1, with four inputs driven by the outputs of the adjacent CLBs, as shown in Fig. 8, and the remaining four inputs driven by other OCWs. For the four OCWs whose QM wire terminates in this tile, a route can be extended to the OCW that starts here through the topmost DP multiplexor. There will also be four DM wires that terminate in this tile, and signals on those wires can muliplex onto the OCW that starts here through either the lower DP multiplexor or the QP multiplexor. The decision to extend the QM wires through only the DP multiplexor is typical of the kinds of engineering trade-offs that must be made when designing a routing architecture. Since the QP line is typically faster than the DP lines, some speed might be gained by increasing the width of the QP multiplexor to allow the QM extensions to connect to the QP wire. However, the additional width of the multiplexor would slow down all the connections through the multiplexor due to the additional loading and would increase the area of the chip. The increased area would directly impact cost and could slightly reduce speed because of the increased lengths of wires. In the end, the decision to design the Mux-C as shown in Fig. 12 was based on place and route experiments that showed that the additional connections would not provide a significant routability advantage.

Another feature of the routing architecture important for difficult routing tasks is the route-thru, the ability to enter and exit a CLB without using its logic resources. This route-thru ability can be used to bypass congested areas and get to an adjacent Mux-C. Because it was designed as a last resort, use of a route-thru is an indicator of insufficient local routing and was used early in the design process to guide the addition of resources. In the final version of the routing architecture, route-thrus are used infrequently.

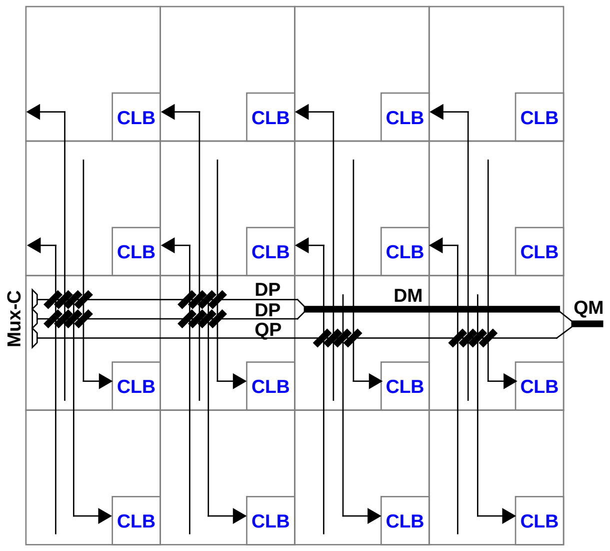

Figure 11. Connectivity between an OCW and nearby EIPs. The short angled marks are PIPs.

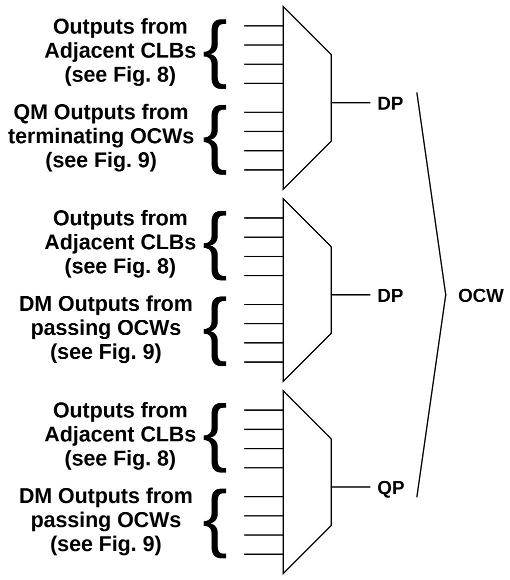
Outputs from Adjacent CLBs (see Fig. 8)
QM Outputs from terminating OCWs (see Fig. 9)
DP
Outputs from Adjacent CLBs (see Fig. 8)
DM Outputs from passing OCWs (see Fig. 9)
DP
OCW
Outputs from Adjacent CLBs (see Fig. 8)
DM Outputs from passing OCWs (see Fig. 9)
QP

Figure 12. Detailed view of a single edge of a Mux-C.

An important feature of the Alexander routing architecture is that every route is buffered. Recall that eliminating the unpredictable delays that result from unbuffered interconnect segments is one of the key ideas behind the Alexander architecture. To accomplish this, every output from the Mux-C and every PIP connecting to an EIP is buffered. As a result every signal is buffered at least twice: once when it emerges from the Mux-C, and once when it enters a logic block via an EIP.

As a result of this buffering, a signal connection that is routed using a Mux-C and an EIP will incur a delay equal to the sum of the delays of the buffers in the Mux-C and the EIP, plus a fixed overhead. We can use this method to compute the delay of any route as long as we know the number of Mux-Cs that the route uses. Note that due to the buffering, this delay is largely independent of the signal fanout.

7. ANALYSIS AND RESULTS

In this section, we analyze the characteristics of the routing architecture in terms of our goals — performance, predictability, routability, and area efficiency — and present specific results from analyses of the architecture that highlight its capabilities. We also discuss the affect of the routability and predictability on the CAD software that must implement users' designs on the FPGA fabric.

We begin with an example route that shows how the three elements of the routing fabric — Mux-C, OCW, and EIP — work together to complete a route. In Fig. 13, the source CLB is in the upper left-hand corner of the array of tiles, and it drives two target CLBs. The thicker lines indicate routing resources that were used to route the signal, leaving the thinner lines and other resources not shown to be used for routing other signals. Tracing the signal in our example, it is first routed from the source CLB to the Mux-C in the adjacent tile. To reach the nearest target, the signal uses a PIP on the double-PIP wire to connect to an EIP and directly into the CLB. This connection is crossbar-like because it requires a single PIP. To reach the far target, the signal must connect to the Mux-C five tiles from the signal's source so that it can turn downwards towards the target. To connect between the two Mux-C elements, the signal uses a double-PIP (DP) followed by a double-metal wire (DM). Note that since the DM is lightly loaded, it is a very fast path between Mux-C components. Once in the second Mux-C, the signal leaves via the OCW heading downwards. In this case, the signal uses the quad-PIP (QP) wire, so that it can connect directly to the EIP and so to the target CLB, completing the route. This example also shows the importance of the fully buffered interconnect. Because of the buffering, the delays to each of the targets are independent of one another and can be computed by simply summing buffer delays. In contrast, in an unbuffered routing scheme, the delays could not be calculated until all the targets had been connected, and the calculation would be a non-linear function of all the resistances and capacitances seen by the interconnect. This example shows not only how the Mux-C, OCW, and EIP work together, but how the buffering provides fanout independence.

The combination of Mux-C, OCW, and EIP also provides excellent routability. High signal routability is ensured by providing extensive routing resources and by adhering to a strict paradigm of PIP placement. The paradigm dictates lower flexibility for a signal to get from a CLB into the Mux-C, but extremely high flexibility for a signal to connect from the routing fabric to another CLB. By ensuring that all signals have ample access to the routing fabric, all

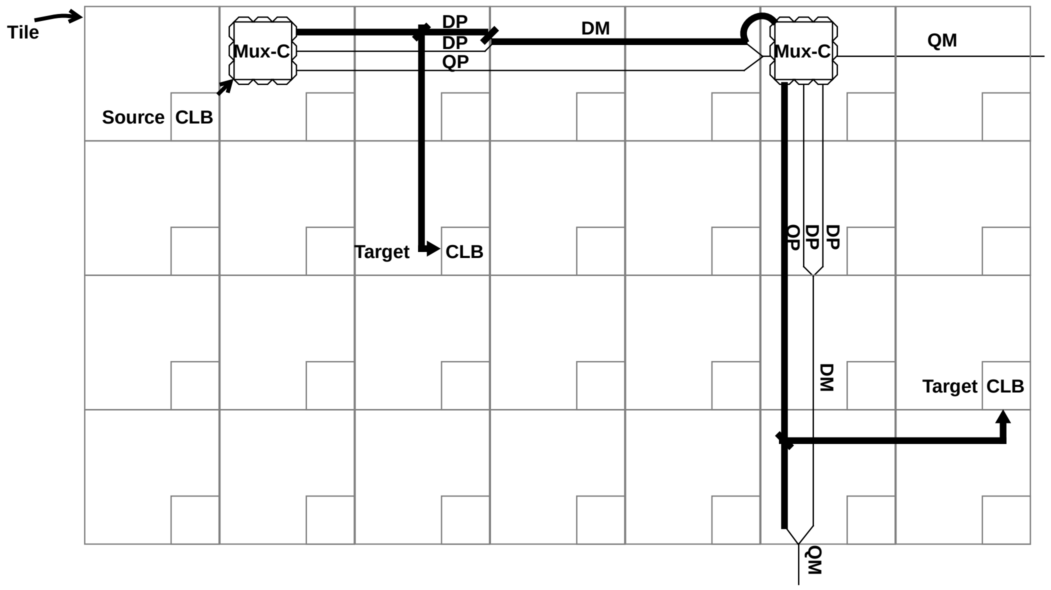

Figure 13. Routing example: this route from the source CLB to the target CLB uses 2 Mux-C components, 2 OCWs, 1 EIP at the input to the target CLB.

connections can be made. One evaluation of routability is given in Fig. 14, which shows the number of single-PIP routing paths from the Y output of the grayed CLB in the center of the diagram to the other CLBs in the diagram. As with the other figures, the grid is the array of tiles that compose the FPGA, and each tile contains a single CLB. The number shown above the CLB in each tile is the number of distinct paths to that CLB from the grayed CLB in the center of the figure. This is an impressive result because this counts only the paths that require a single PIP, so there are many more available paths that require more than one PIP. The single-PIP paths are important because a single PIP connection looks like a crossbar and has less interconnect delay. This result shows how the combination of Mux-C, OCW, and EIP provide abundant fast routing resources on the FPGA.

The Alexander Architecture is a high performance architecture. This comes mainly from the ability of the OCW to route signals within a distance of four tiles with very low delay and to extend signals in increments of eight additional tiles with minimal incremental delay. The fully buffered interconnect also eliminates the high RC buildup due to fanout. The routing fabric gives a high degree of connectivity enabling signal connections to span several tiles while the buffering ensures that the connection delays remain largely independent of the distance that they span and the fanout of the signal. To illustrate this point Fig. 15 plots a map of the direction and distance that a signal can travel with respect to the delay that it incurs. In Fig. 15 we consider the unit of delay as the delay incurred in passing through a single Mux-C. Regardless of the number of Mux-Cs traversed, every signal will incur additional fixed delays entering the first Mux-C from the driving CLB and entering the load CLB via an EIP.

Notice the gradation of delay that occurs with distance from the source. There are two important facts to note here: the slow and predictable degradation of delay as the distance from the source increases, and the discrete steps in which the delay increases. This predictability comes from the architecture's two significant features: the connectivity of the routing fabric and its buffering scheme.

Using a similar methodology used in the construction of Fig. 15 we can generate tables that can be used to estimate delays. Accurate delay estimates enable better evaluation of design trade-offs during manual logic design or automatic logic synthesis. This approach also results in very accurate delay estimates during the placement phase of the implementation of a user's design on the FPGA. Further, since the delay estimates can be made using simple table lookup the delay estimation is considerably faster than traditional RC tree analysis methods. These characteristics greatly improve the efficiency and ease of writing software tools to implement users' designs on the Alexander FPGA.

Although experimental software was developed to evaluate alternatives as the Alexander architecture evolved, this software was not tuned for production, nor were the conservative timing values used for experiments optimized to match silicon. For these reasons, timing results from placed and routed designs are open to a certain amount of interpretation, and we refrain from quoting detailed results here. We consider it a validation of the architecture that 97 of 114 designs from a typical design suite had better clock performance when placed and routed with experimental Alexander software than when placed and routed with the

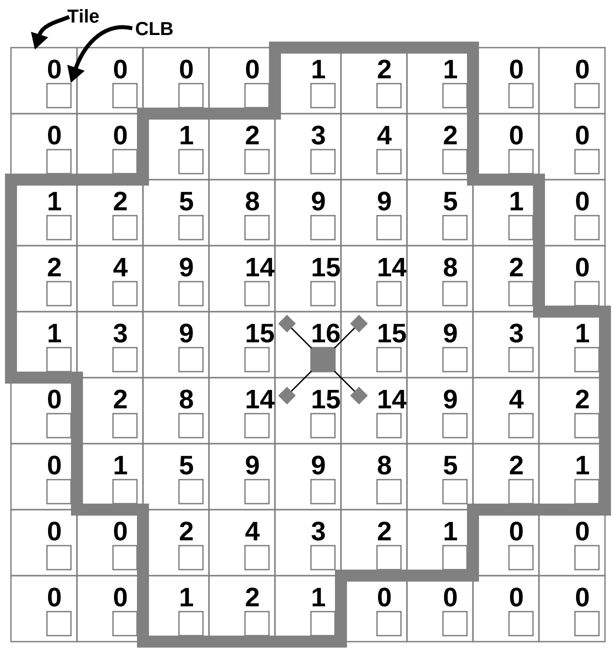

Figure 14. Number of available routing paths that use only 1 PIP from Y output to F input of nearby CLBs.

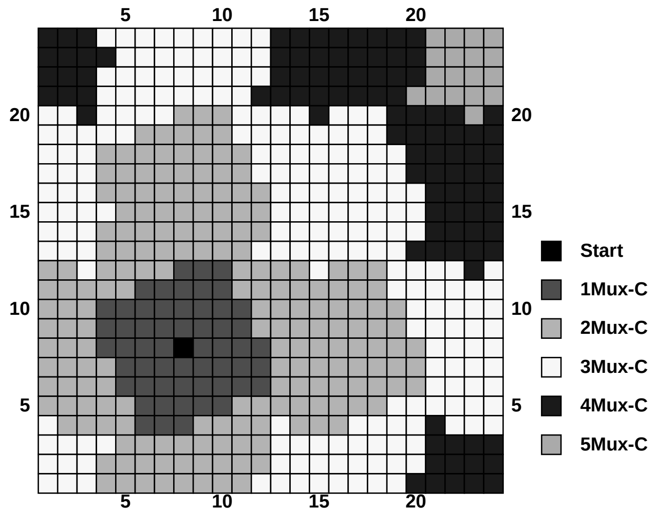

Figure 15. Reach of the Alexander routing fabric. Mux-Cs required to reach a given CLB from the start CLB.

production software for a XC4000E in the same fabrication technology.

Even with the somewhat reduced confidence level in the timing data, the fanout independence of the architecture is clearly validated by the graph in Fig. 16. This graph shows maximum clock frequency for a family of special test designs placed and routed with experimental Alexander software and production XC4000E software. These test designs were a single flip-flop driving a large number of other flip-flops. The graph shows that as the fanout to the load flip-flops increases, the maximum achievable clock frequency on the XC4000E degrades substantially while the clock frequency for Alexander remains largely constant.

To understand better several aspects of the Alexander FPGA, complete layout and fabrication was undertaken. The CLB tile in the Alexander architecture was less than 40% larger than the XC4000E tile when laid out in the same process. However, this increased area was paid for not only by increased speed but by increased routability over the XC4000E. The Alexander project culminated with the creation of the completely functional test chip shown in Fig. 17. Many of the key innovations from the routing fabric and other aspects of the architecture are leveraged in the Xilinx XC4000EX, XC4000XL, XC4000XV families and in the recently released Virtex family.

8. CONCLUSIONS

The Alexander routing architecture provides fast, predictable routing interconnect. The architecture is woven together with an interconnection of Mux-Cs, OCWs, and EIPs, giving the architecture the flavor of a cross-bar. The performance is attained by reducing the number of PIPs required to reach a destination. The predictability is a result of the fanout independence of the buffered interconnect, as well as the predetermined number of interconnections required for routing signals between any logic blocks in the array. This architecture lends itself to an efficient silicon solution that integrates well with place and route design tools, to provide a vehicle for high performance FPGA design.

9. REFERENCES

[1] K. Pierce, C. Erickson, C. Huang, and D. Wieland, “Interconnect Architecture for Field Programmable Gate Array”. United States Patent No. 5,581,199. December 3, 1996.

[2] W. Carter, K. Duong, R. H. Freeman, H. Hsieh, J. Y. Ja, J. E. Mahoney, L. T. Ngo, and S. L. Sae, “A User Programmable Reconfigurable Gate Array,” *Proc. 1986 Custom Integrated Circuits Conference*, May 1986, pp. 233-235.

[3] *The Programmable Logic Data Book*, Xilinx Inc., 1996.

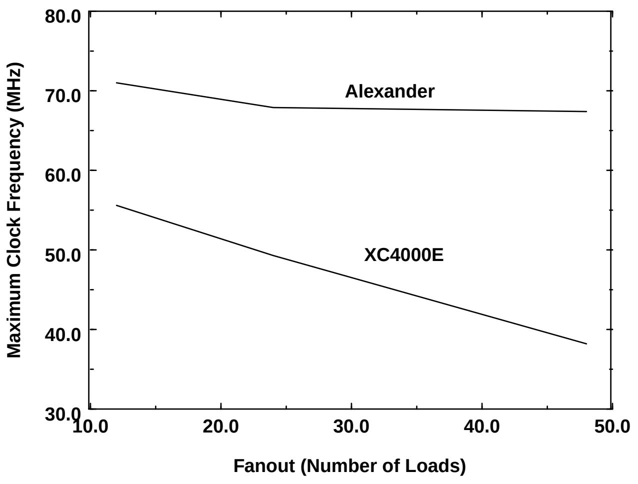

Figure 16. Maximum clock frequency comparison for high-fanout nets

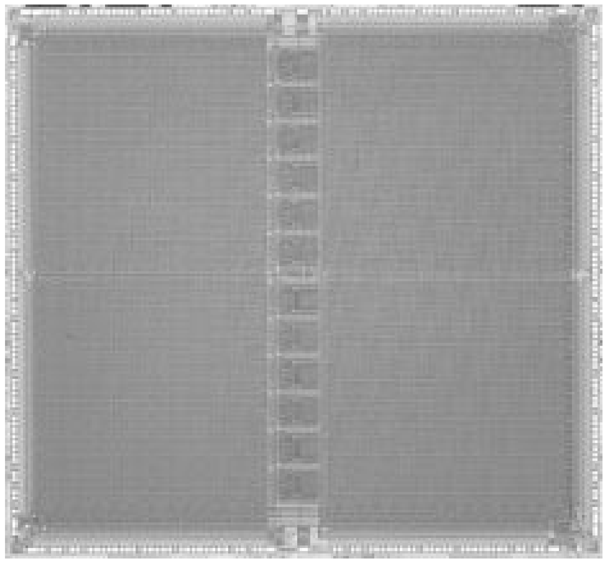

Figure 17. Photograph of fabricated test chip.