Statistical Analysis and Design of HARP Routing Pattern FPGAs

Gang Wang, Satish Sivaswamy, Cristinel Ababei, Kia Bazargan†, Ryan Kastner* and Eli Bozorgzadeh

***Abstract*— Modern FPGA architectures provide ample routing resources so that designs can be routed successfully. The routing architecture is designed to handle versatile connection configurations. However, providing such great flexibility comes at a high cost in terms of area, delay and power. We propose a new FPGA routing architecture that utilizes a mixture of hardwired and traditional flexible switches. The result is about 30% reduction in leakage power consumption, 5% smaller area and 20% shorter delays, which translates to 25% increase in clock frequency. Despite the increase in clock speeds, the overall power consumption is reduced.**

I. Introduction

Prohibitive ASIC mask costs and stringent time-to-market windows have made FPGAs an attractive implementation platform in recent years. However, circuits implemented on FPGAs are typically slower, occupy more area, and consume more power than ASIC circuits [25]. The FPGA routing architecture is the main culprit in making FPGAs worse than ASIC chips in area, delay and power; a typical FPGA routing architecture uses about 70-90% of the total transistors on the die [6].

A significant body of work from the past two decades focused on switch box design and segmented routing architectures. The basic idea is to use highly flexible switches where horizontal and vertical tracks meet, to facilitate all possible connections between the adjacent tracks. A sketch of the disjoint switch box is shown in Figure 1.

Assuming all tracks have unit length, the disjoint switch box (see Figure 1) can route a large subset of possible routing trees to connect the terminals of a net. As a result, the overall channel width will not be high. However, flexibility in routing comes with great performance costs. Building a routing tree from many segments that are connected by switches has the following disadvantages:

- **Circuit Delay:** The delay of a net is mainly dependent on the number of programmable switches in its routing path [14], [4]. Hence, a large number of programmable switches contributes greatly to the overall circuit delay.

G. Wang and R. Kastner are with the Department of ECE, University of California at Santa Barbara, Santa Barbara, CA 93106, Email: {wanggang,kastner}@ece.ucsb.edu

S. Sivaswamy and K. Bazargan are with the Department of ECE, University of Minnesota, Minneapolis, MN 55455, Email: {siva0024,kia}@umn.edu

C. Ababei is with Magma Design Automation Inc., 5460 Bayfront Plaza, Santa Clara, CA 95054, Email: cababei@Magma-DA.COM

This work was supported in part by a grant from NSF under contract CAREER CCF-0347891

E. Bozorgzadeh is with the Computer Science Deparment, University of California - Irvine, Irvine, CA 92697, Email: eli@igor.ics.uci.edu

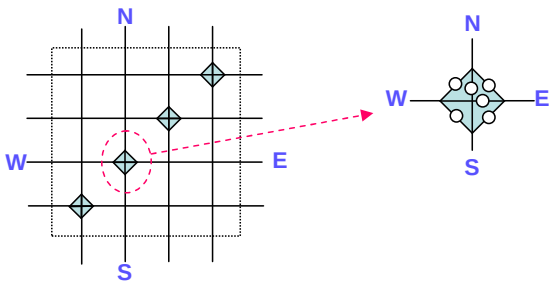


Fig. 1: SRAM-based Switch Box.

- **Area:** By increasing the number of programmable pass transistors (which correspond to the small circles in the switch on the right in Figure 1) inside each switch, we pay an area penalty as each of the programmable pass transistors requires an SRAM cell for programming it and possibly buffers to improve signal slew.
- **Leakage Power:** Leakage power is becoming a major component of the total power consumption [1] and the majority of the leakage power consumption in FPGAs occur in the routing switches [7].

A. Related Work

There has been a lot of work on programmable architectures to improve the performance of FPGAs. Modern FPGAs utilize multi-length horizontal and vertical segments. Recently, there has been a flurry of research in structured ASIC solutions [24], which aim to provide a middle ground between ASICs and FPGA. Tong *et. al.* [18], Jayakumar and Khatri [11], and Yan and Marek-Sadowska [22] proposed via-configurable gate array implementation platforms, in which connections are programmed by the presence or absence of vias. This results in improvements in power-delay performance but the flexibility and cost savings are limited because of its mask programmability.

There have been several CAD techniques aimed at reducing the number of "bends" in the routing architecture. For example, the work in [12], [13] focused on the concept of using prespecified patterns to route a net. By doing so, one would allow a more accurate prediction mechanism for metrics such as congestion and wirelength earlier in the design flow. The work in [12], [13] focused on an ASIC design flow, rather than the FPGA flow found in this paper.

Maidee *et. al.* [15] proposed a "terminal alignment" heuristic, which reduces the number of bends on nets, and hence eliminates switches that need to connect horizontal tracks to

vertical ones. As a result, the number of switches used in routing of critical nets decreases. They achieved 5% delay improvement over VPR using a simulated annealing engine.

B. The Idea of HARP (HArd-wired Routing Pattern) FPGAs

In this work, we extend the idea of eliminating bends and switches (inspired by FPGA architectures such as Xilinx's Virtex, and placement and routing tools such as [12], [13] and [15]) to two dimensions; instead of just hardwiring two horizontal or two vertical segments to form longer wires, e.g. segmented routing architectures such as Xilinx Virtex, we form hardwired junctions between horizontal and vertical segments inside switch boxes. These junctions create routing segments in the shape of T's, L's and +'s and their rotated versions. An example of such a switch box is shown in Figure 11. As a result of hardwiring connections, we eliminate some programmable switches, which decreases the delay, area and power dissipation. However, we must be careful that the reduction in programmable switches does not severely affect the routing flexibility.

Figure 2 shows a conceptual diagram of a HARP architecture that contains 6x6 switch boxes (logic blocks are not shown in the figure). The figure shows a subset of the routing resources. Switch box 4D contains L and T HARP routing resources and one traditional flexible switch (the bottom track).Two of the L connections span one switch box on each side whereas the other two span two switch boxes on each side. Switch box A1 shows an L connection that spans 5 switch boxes on each side. The logic block at tile 6A can connect to the logic block at tile 1F through the fast L connection. An L HARP resource and a T resource are merged in tile 6E to form a more complex HARP pattern. Note that this connection is hardwired, as opposed to the connection between the flexible switch and the T HARP resource at tile 4D.

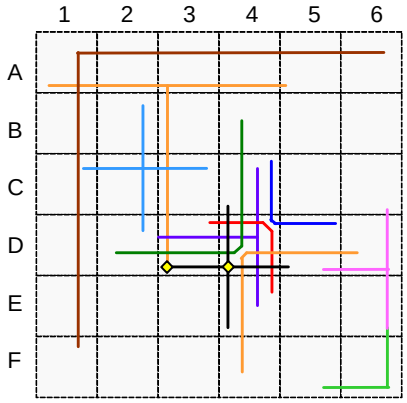


Fig. 2: Global view of a HARP architecture.

Our contributions can be briefly described as follows:

- *Routing requirement analysis:* A number of circuits are placed and routed on traditional FPGA architectures, and the routing patterns that are formed in the switch boxes are analyzed.
- *HARP architecture generation:* based on the frequencies of the patterns that the router uses, we instantiate hard-wired routing patterns (HARPs) that replace some switches in the routing structure.
- *Placement and routing with HARPs:* We place and route circuits on the new HARP architecture.

After placement and routing on HARP architectures, we report results of area, delay, power and channel width. The experiments show that our technique maintains the programmability of FPGAs, while improving their performance metrics.

The rest of the paper is organized as follows. In Section II, we describe the terminology that we use throughout the paper and the overall flow of our architectural design. In Section III, we perform an empirical analysis on detailed routings to find the most common routing patterns and their densities. Based on this analysis, we design the architecture of the switch boxes containing the hard-wired routing patterns. Details of this step are discussed in Section III-C. Section IV explains how hard-wired routing patterns inside the switch boxes are exploited by the router. Experimental results are presented in Section V. Section VI details a method that generates a regular HARP FPGA and presents the results of placement and routing on such architectures. We conclude in Section VII, by outlining our main contribution and discussing future research directions.

II. Preliminaries

A. Basic Terminology

The routing of a multi-terminal net is frequently modeled as a *rectilinear Steiner tree (RST)*. A RST has three types of *joint patterns*: L-shape, T-shape, and +-shape. An FPGA routing architecture with uniform unit-length segmentation has a switch at each of the joint patterns. Additionally, there are switches for horizontal (H) and vertical (V) routes that span more than one channel. Modern FPGA devices use multi-length segments ($-$-shape and $|$-shape) in order to reduce the number of switches along the horizontal or vertical routes of the nets. This enhances the delay of the routing; however, it reduces the flexibility of the architecture.

We call the switch shown on the right side of Figure 1 a *flexible* switch. A multi-length segmented architecture merges the "W" track and the "E" track to form a longer segment. This is equivalent to removing the pass transistors (and their associated SRAM cells and buffers) that connect the "E" or "W" tracks to other tracks. The result is a hardwired connection between "E" and "W". The area of this new switch is smaller, however, it is less flexible than the original *flexible* switch. If we allow the horizontal track to also connect to the vertical track at this junction (*e.g.*, the way hex lines in Xilinx architectures connect to other segments on the middle point), then two more switches will be used to provide connectivity between wire segments "WE" and "N", and also between "WE" and "S". Obviously, the area and delay of this switch increases, but we gain flexibility.

To the best of our knowledge, no one has extended the idea of hardwiring pass transistors to junctions that are formed between horizontal and vertical tracks. In this work, we study *HARP* (HArd-wired Routing Pattern) architectures that utilize hardwired "switches" at certain junction patterns. The three

joint patterns (L, T, +, and H/V) and their various orientations result in eleven possible HARPs: $\top$,$\perp$, $\dashv$, $\vdash$, $\ulcorner$, $\urcorner$, $\llcorner$, $\lrcorner$, $|$,$-$ and $+$.how to selectively decrease the flexibility of the routing architecture

B. HARP-based FPGA Routing Architecture Design Flow

Figure 3 shows our design flow to introduce HARPs into *traditional* FPGA routing architectures. First, we place and route a number of circuits on a traditional FPGA architecture. By analyzing the routes of the circuits, we extract the frequency at which different HARP patterns are used in switch boxes. Next, we use the results of the pattern distribution analysis to create a new architecture that has a mixture of flexible and HARP switches. Finally, we place and route designs on the new HARP architecture and compare the results with the traditional architectures.

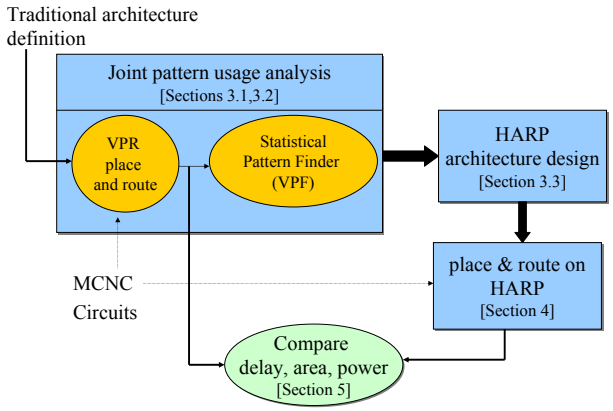


Fig. 3: HARP-based Routing Architecture Design Flow.

III. Routing Pattern Analysis

In this section, we discuss the statistical analysis of routing pattern frequencies and correlations between circuits, architectures and these patterns.

A. Testing Benchmark and Routing Result Generation

After placing and routing the MCNC benchmark circuits, we observed how often each of the joint patterns can be found in a route of each net. Note that the placement and routing algorithms will affect the frequency of these patterns. We will discuss this issue in Section VII.

Statistical information can guide us on how often we need to replace flexible switches with HARP resources inside switch boxes. To find the pattern frequencies, we routed all the benchmarks on a given traditional segmented FPGA routing architecture applying the VPR FPGA place-and-route tool [3]. For a given routing segmentation architecture, we routed each circuit, detected the patterns in the route files, and applied statistical analysis on the data.

In our studies, we considered two segmentation architectures: unit-length segmentation and multi-length segmentation (similar to Xilinx's Virtex family). We used the VPR router in three different modes: Timing-driven, Routability-driven without bend-cost, and Routability-driven with bend-cost. We experimented our technique on the MCNC benchmark suite [23].

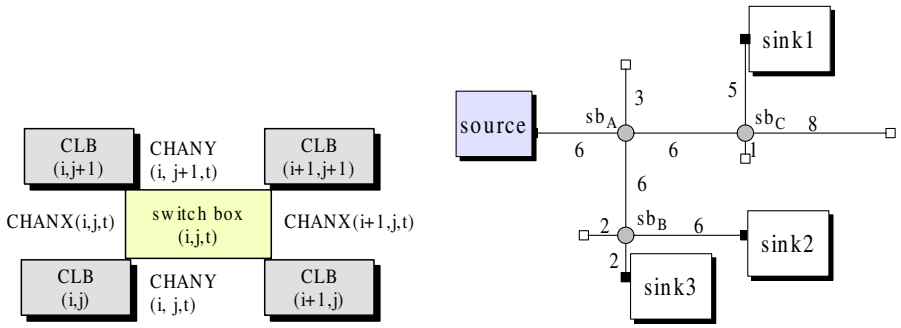
B. Analysis of Routing Patterns

1) VPR Pattern Finder Tool: In order to analyze the behavior of the routing patterns, we have implemented *VPR Pattern Finder*(VPF), a graphic tool for parsing, visualizing and analyzing the VPR routing results [19]. VPF takes a VPR routing result file as input and automatically extracts the routing information, identifies the connection patterns at switch points, and in turn generates statistical reports for different patterns.

In our analysis, we focus on the connection patterns found in switch boxes. For a given FPGA layout, a switch point is indexed by a tuple (i, j, t) as shown in Figure 4(a), where i and j indicate the physical location of the switch box containing that switch point and t identifies the track being used. Based on the structure of the switch point, we have 11 possible connection patterns. They are $\top$, $\perp$, $\dashv$, $\vdash$, $\ulcorner$, $\urcorner$, $\llcorner$, $\lrcorner$, $|$, $-$ and $+$. As an example, Figure 4(b) shows a sample net, which contains only one source and three sinks. Empty rectangles show ends of the segments that are not connected to this net. The numbers on the lines denote the length of the connections (that are possibly formed by connecting two or more segments). Based on the above discussion, switch box sb_A has pattern $\top$, sb_B has pattern $\vdash$, and sb_C is of pattern $\lrcorner$.

It is important to find out how each HARP extends along different directions - the *pattern length*. This information can provide more insight into the routing behavior and offer useful guidance for improving routing quality by hard-wiring these patterns. VPF performs this analysis using the following simple algorithm: (i) For each marked switch box, identify its pattern. Based on the pattern information, try to trace along the valid directions starting from the switch box; (ii) Stop when we meet a switch point that has a pattern other than $|$ when we are tracing vertically or $-$ when we are tracing horizontally. Furthermore, we need stop the tracing when we reach the source or a sink; (iii) Report this distance as the result of the current direction; (iv) Take the minimum among all directions as the pattern length of the current switch point. Following the same example shown in Figure 4(b) and assuming the numbers by the segments indicate the segment lengths, switch points sb_A, sb_B and sb_C have pattern lengths 6, 2 and 5 respectively.

2) Statistical Results and Analysis: Statistical information about the switch box patterns obtained from VPF provide insight into the behavior of the placement and the routing tools as well as resource demands of the circuits. Figure 5 shows the normalized pattern distributions (in percentage of all the switch points) for different benchmarks and segmented architectures. Among them, the unit-length (*D*) results are generated on an architecture that only supports segment of length one, while those of *A*, *B* and *C* are generated on a Virtex style architecture with multi-length segment routing architecture. *A* is generated with the routability-driven routing without bend



(a) switch box indexing (b) pattern labeling

Fig. 4: Switch box indexing and pattern labeling in VPF

cost, *B* corresponds to routability-driven with bend cost, and *C* is generated using the timing-driven routing mode. Placement for all experiments was done using the timing-driven mode. Routability-driven algorithm aims at minimizing the length of each route. When no bend cost is considered, many bends can appear in the routes. On the other hand, the router considering the bend cost generates the routes with smaller number of bends. Timing-driven router does not consider the bend cost directly. Since the delay of a route in FPGA is dominated by the number of switches and each bend includes a switch, the algorithm avoids generating many switches. As a result, each of the three routing algorithms manages the bends in the routes differently. We applied the three algorithms to observe the distribution of all different patterns when different routers are used.

One interesting observation that can be made from Figure 5(a) is that the multi-length segmented architecture greatly changes the pattern distribution compared to unit length. The combined frequency of vertical and horizontal patterns drops from 63.3% to 41.2% [1]. On the other hand, as seen in Figure 5(b), when using different router settings on the same Virtex style architecture, the results do not vary much. In other words, the architecture seems to have a much bigger impact on the switch box pattern distribution than the routing algorithm.

Figure 5(a) shows there is little change in the percentage of the T patterns or the + pattern when we switch from unit-length to the multi-length segment architecture. On the contrary, there is a significant increase for the L patterns . The combined frequency of all the L patterns increases from 27.46% for the unit-length architecture to 41.62% for the multi-length architecture. In other words, for the multi-length architecture, the possibility of having an L patterned switch point is comparable to (if not more than) that of a vertical or horizontal pattern. This is a notable difference compared to the unit-length results, in which the vertical and horizontal patterns overwhelmingly dominate the pattern distribution.

[1] The reason is that the multi-length segments are in essense horizontal and vertical connection patterns. For example, a horizontal segment of length 6 is the equivalent of 5 horizontal switch connections in the single segment architecture.

Next, we analyze the length of the patterns using the method discussed in Section III-B.1. Figure 6 illustrates the pattern length distributions for our testing benchmarks. The x-axis in these graphs is the pattern length, while the y-axis is the normalized percentage for switch point patterns with the given length. Benchmarks A, B, and C are generated using the same multi-length Virtex style architecture with different routing considerations, i.e. routability-driven without bend-cost, routability-driven with bend-cost,and timing-driven; Benchmark D is the timing-driven result using the unit-length architecture.

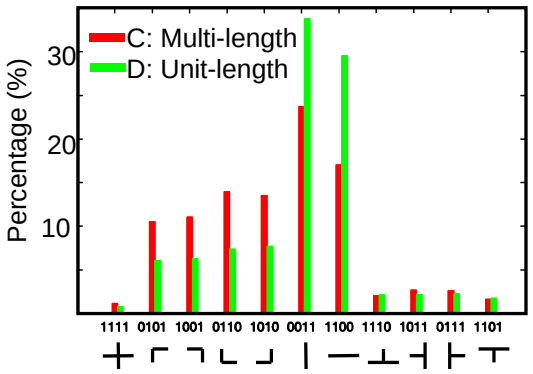
We can observe that all these graphs share some common characteristics. First, for the unit-length architecture, the pattern length distribution drops rapidly and monotonically as the length increases. The results for the multi-length architecture are not monotonically decreasing but still follow a similar trend except for length 6, which shows spikes on all patterns. On all patterns except +, there is a small spike at length 12 too. Such harmonic behavior demonstrated by these spikes is not surprising because in the multi-length architecture, the majority of the segments are of length 6, where switch connections are allowed at both ends of the segments.

We also performed further analysis (results not shown in this paper), focusing on the geometric distribution of different patterns. We observed uniform distribution of all patterns with the exception of one benchmark[2]. It is important to note that our results are valid only for the class of circuits represented by the MCNC benchmarks that we used in these studies. For example the routing pattern statistics of data-path designs might show different characteristics than those shown here. A bus-based CAD flow such as [26] can be used to place and route such designs, and the routing analysis can be used to generate HARP architectures tailored to data-path circuits.

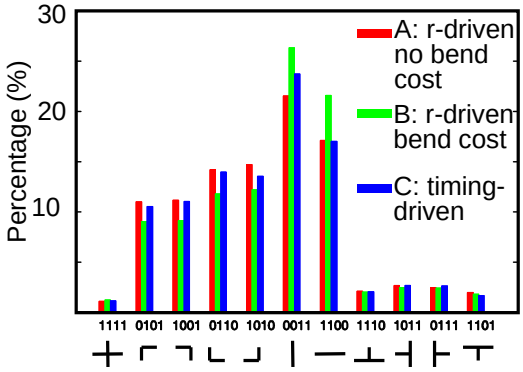
C. Architecture Design

Architecture design for FPGAs is a complex problem and much work has been done in this area since FPGAs were

[2] For circuit "bigkey", the + patterns were concentrated in the two horizontal channels at the center of the chip.

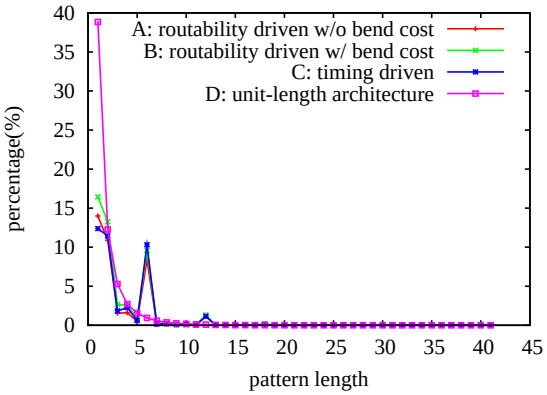


(a) results for different architectures (both timing driven)

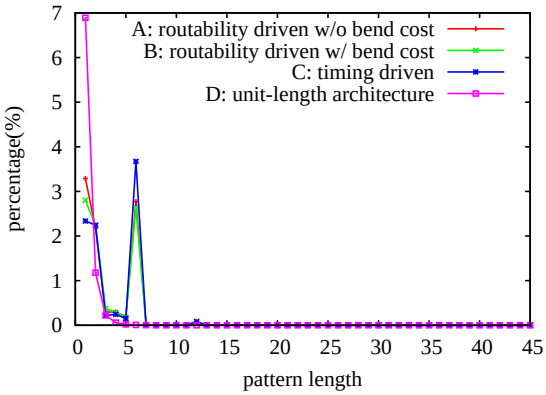


(b) results for different routing considerations on the same multi-length architecture (Virtex-like)

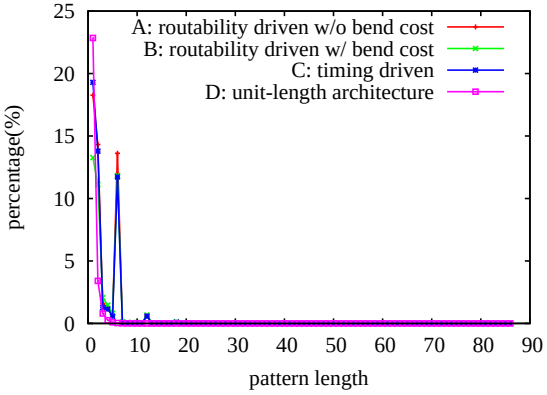
Fig. 5: Switch box pattern distributions



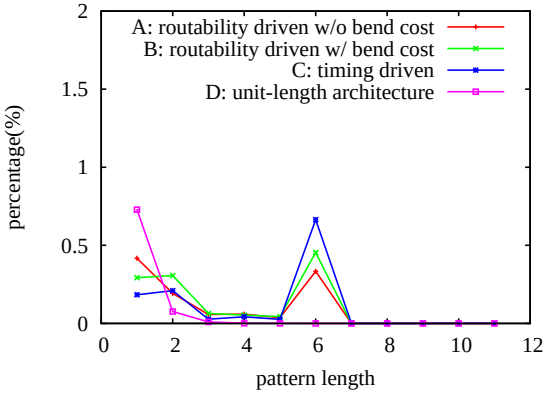
(a) Length distribution for pattern | and −



(b) Length distribution for T patterns



(c) Length distribution for L patterns



(d) Length distribution for pattern +

Fig. 6: Distribution statistics on switch point pattern lengths

first proposed. There are many architectural factors (such as switch-block or switch-matrix style, switch-block flexibility F_s, connection-block flexibility F_c, frequency of switch-blocks along routing segments, channel segmentation and staggering, clustering of LUTs in CLBs), which contribute to the quality of the final FPGA platform. More details about such architectural features is provided in our FPGA'05 paper [27].

Based on the analysis presented in the previous section, we design a new switch box, which will include hard-wired routing patterns. This means that we remove a certain number of programmable switches (derived from statistical analysis of the routing profiles of various circuits) from the switch boxes and replace them with wires. The composition of these hard-wired patterns are chosen after careful analysis of the routing profiles, hence the effect on the routability of circuits is minimized. We have experimentally verified the minimum effect of HARP resources on the routing of circuits (see Section V).

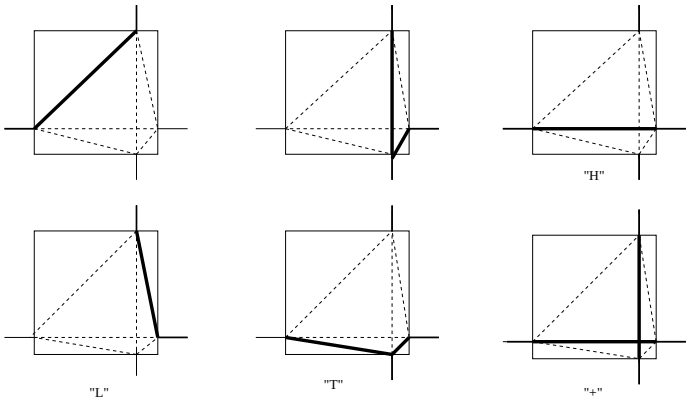


Fig. 7: Some possible hard-wired patterns

Figure 7 shows some of the possible HARPs that can be present inside the switch boxes. The hard-wired patterns are shown using solid lines indicating that they are wires and not programmable switches. The next section describes how the routing tool is made aware of these patterns and how they are exploited to reduce the delay, area and power dissipation.

IV. Routing with HARPs

To harness the advantages of HARP architectures, the placement and routing tools must be adapted to use hard-wired resources for timing critical nets and only use regular switches where hard-wired resources are not available.

In this work, we would like to fully exploit the hard-wired patterns present inside our switch-blocks. This can be done in the detailed routing stage by constructing a routing graph with the hard-wired routing patterns embedded as low cost edges. VPR [2], and the power model from Wilton, *et. al.* [16], which we use for our work employ a routing graph construction approach to perform detailed routing. The routing segments and the logic block input and output pins are represented as vertices in the routing graph with a certain cost associated with them. Edges in the routing graph correspond to the connections between them. Edges may be bidirectional or unidirectional depending on whether a pass-transistor or a buffered switch is used [3]. A sample routing graph is shown in Figure 8 [3].

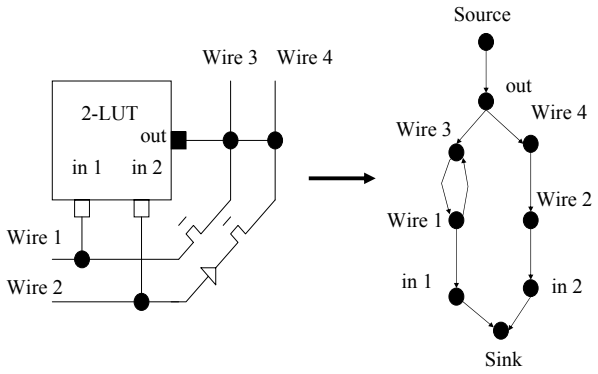


Fig. 8: Sample routing graph

The way the routing graph is constructed changes with the presence of hard-wired patterns. These changes occur inside the switch boxes. Figure 9 shows the routing graph for a disjoint switch box (with pass transistor switches) with all tracks terminating at the switch box, and a disjoint switch box with a L-shaped hard wired pattern embedded in it. With the L-shaped pattern in the switch box, the routing graph contains only those edges forming the pattern and all the other edges are removed from the graph. For instance, in figure 9, edges between nodes (A,C), (A,D), (B,C), (B,D) are removed from the routing graph since they do not participate in forming the patterns.

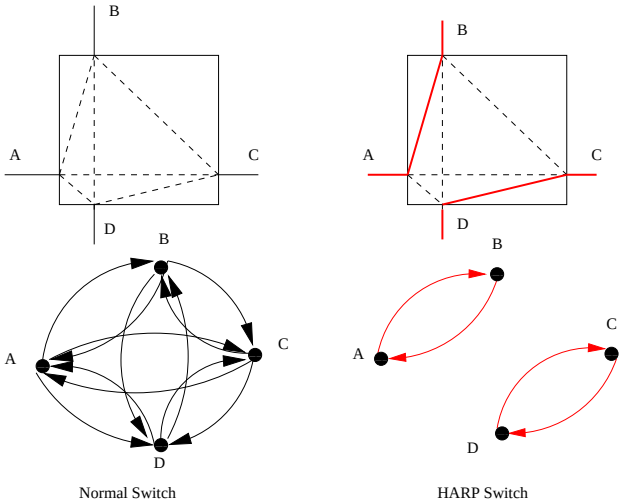


Fig. 9: Routing graph with HARPs

Based on the results of the analysis presented in Section III, we first determine the number of different HARP patterns that need to be inserted inside the switch boxes. Next, the FPGA chip is scanned row-by-row and patterns are inserted based on their desired percentages. When introducing these patterns, we make sure not to connect different hard-wired patterns together

to form large trees. The reason for this restriction is illustrated by the example of Figure 10.

When we use two adjacent hard-wired patterns, an L-shaped pattern and a T-shaped pattern to connect terminal *A* to terminal *B* in the figure, a dangling segment is formed. This is undesirable as it adds extra capacitance and resistance, which is contrary to the goal of reducing overall power and delay. This problem is overcome by making sure that not many hard-wired patterns are connected back-to-back. In the rest of this section, we present our algorithms assuming that *no* two HARPs are allowed to connect back-to-back, but later on in Section V, we relax this restriction a little and observe that the *limited* use of merged HARPs will improve the quality of the circuit.

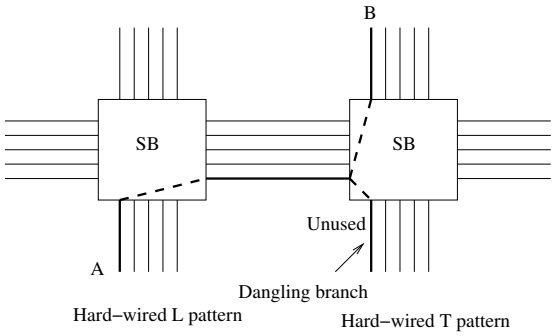


Fig. 10: Connecting Hard-wired patterns together

Once we know the number and location of these hard-wired patterns, we change the way VPR constructs the routing graph and include only those edges (corresponding to wire segments) that are actually connected to the pattern. These edges are inserted as low cost edges so that the router will automatically choose these hard-wired patterns when performing detailed routing. The cost is calculated based on the lumped resistance and capacitance of the wire segment (including HARP and regular segments) connected to a switch. Interested readers can find the pseudo-code for inserting the hard-wired patterns in the architecture in our FPGA paper [27].

Note that ⌜ and ⌟ can be combined into one switch configuration which makes two disjoint connections (one between right and bottom segments, and the other between top and left segments). The same is true with ⌝ and ⌞. See Figure 11 for examples of L patterns (in SB2, the fourth switch from the bottom is an {⌜, ⌟} switch). Our architectural generation code is available for download from [8] for non-commercial use.

A. Estimation of Delay, Area and Power

We use the delay and area models in VPR and the power model developed by [16] to estimate the circuit delay, total area of the chip and the total power dissipation after inserting the hard-wired resources. VPR uses an Elmore delay model to estimate the delay of every net. In this model, pass transistors are represented as resistors and diffusion capacitances to ground. Pass transistors add parasitic capacitance to the wire irrespective of whether they are on or off leading to a higher delay [3]. In our hard-wired resources, we eliminate the pass transistors and replace the resistance and capacitance values of the pass transistors, used in the delay model, with those of the metal wire (of segment length 1).

To accurately determine the delay of using a hard-wired resource, the capacitance of all the segments forming the pattern are included in the total capacitive load being switched. In addition, when only some of the segments of a hard-wired switch are used to route a signal, the remaining segments are made unavailable to route other nets. This avoids potential resource conflicts that could occur when different nets try to use different parts of the same hard-wired switch. For example, when only the vertical segment of the hard-wired L is used to make connections and the horizontal part is dangling. In this case, the horizontal segment is invalidated so that it is not available for use by other nets. More details of this procedure can be found in our FPGA paper [27].

The area model in VPR is based on counting the number of transistors required to implement the FPGA architecture. It reports area in terms of the number of minimum width transistor areas required to implement the circuit on the FPGA [3]. For our hard wired resources, we use the same procedure and count the total number of transistors in our implementation. We use the power model developed by [16] to estimate the total power dissipation. Leakage power is estimated by counting the number of unused transistors and SRAM cells and multiplying them with their individual leakage power. Dynamic power is dependent on the charging and discharging capacitance and the clock frequency, which is the critical path delay. The short circuit power is taken as 10% of the dynamic power. The charging and discharging capacitance is obtained from the parasitics used in the delay model of VPR.

V. Experimental results and analysis

A. System Performance Improvements

We inserted the hard-wired patterns in the switch boxes and used a multi-segment routing architecture with routing-segments of lengths 1, 2, 6, and long lines. The distribution of the segments in each channel are 8%, 20%, 60% and 12% respectively (similar to the Virtex architecture) for all simulation experiments. HARPs were not inserted on long lines, though. We placed and routed the 20 MCNC circuit benchmarks of the VPR package and 3 of the large benchmarks of the Altera QUIP tool set[3] (oc_wb_dma, oc_mem_ctrl, oc_des_des3perf, which have 9872, 8611 and 38,218 CLBs, and 9654, 8726, and 38,452 nets respectively) on HARP architectures and report the results of circuit delay, area, leakage power, total power dissipation and channel width. It is important to note that the reported area is only the transistor count. A complete architecture generation flow would use an ASIC CAD tool to generate a detailed placement and routing of the FPGA architecture itself. Since the layout is going to be generated automatically, the area efficiency of the HARP FPGA would probably be worse than a traditional FPGA that is manually layed out. However, our tile-based HARP architecture (will be presented in Section VI) would enable designers to manually

[3]Since these circuits make extensive use of register enables and other control signals that are not present in VPR, we recompiled the designs in Quartus by suppressing the use of adders and multipliers.

lay out a limited number of tiles and replicate them throughout the chip.

We updated the delay look up tables used by the placement tool of VPR to reflect delays of HARP connections. However, this had only a marginal impact on the placement quality, more specifically, the delay improved 1-2% which is statistically insignificant, but channel width increase by about 5% and consequently the area and the total energy consumption increased by 3% and 5% respectively. The reason is that these delay lookup tables are built assuming no congestion is present, and hence the best routing resources for delay are always available. This is an optimistic lower bound on the routing delay between two points. In reality, the router will have to use traditional, slower switches for some nets due to congestion.

[—DelayTableExplanation—] The problem is compounded by the fact that by using HARP resources, the delay difference between timing critical and non-critical nets reduces. When more nets become critical, congestion becomes a problem as many nets try to occupy the relatively few HARP resources. The resulting congestion increases the final channel width and worsens other metrics but the worst case delay remains more or less the same. Since we do not have a good estimate of the congestion at the placement level, we decided to leave the delay tables untouched to better capture the lack of enough routing resources at the routing level. We plan to improve the placement quality by using better models of the HARP architecture and use a tightly coupled timing analyzer in our future work.

Results of the execution of VPR with 50% of all switches replaced with HARPs and that of the traditional "Virtex-like" architecture is presented in Table I. Columns labelled "Vtx" show the results of the traditional "Virtex-like" routing architecture. The last row of the table shows the ratio of the geometic mean values for Vtx and HARP.

We observe that the insertion of hard-wired routing patterns has a profound impact on delay and leakage power dissipation, reducing delay by about 17.45% and leakage power by 22.11% on average. Insertion of hard-wired routing patterns as low cost edges in the routing graph encourages the routing tool to use them whenever possible. This leads to a considerable speed up of the circuit. Also, the elimination of the program bits results in fewer SRAM cells and a lower leakage power dissipation. We find that the total area of the circuit decreases by about 5% on average. However, the average channel width increases by around 16.66%. This is expected, since, the introduction of hard-wired routing patterns reduces the flexibility of the routing architecture causing the router to use more tracks to route certain connections. However, the overall routing area of the circuit decreases because the reduction in individual switch area dominates the increase in number of switches caused by increased channel width.

Total power dissipation reduces on average by about 4%. In spite of the 22% reduction in the leakage power dissipation, the total power reduces by only around 4% on average. This is explained by taking into account the dynamic power dissipation. Dynamic power dissipation is dependent on the switching rate of the circuit. With hard-wired patterns in the switch boxes, the critical path delays of the circuits are reduced considerably resulting in a higher clock rate. This causes increased dynamic power dissipation. A fair comparison metric between the Virtex-like routing architecture and HARPs would probably be the power-delay product. We can explore different configurations by considering the energy dissipation or the power-delay product. Power-delay of HARP is 21.5% better than Vtx. Depending on whether optimizing for speed or power is more critical, we can clock the circuits at a higher clock frequency to get a faster circuit or we can clock the circuit at a lower speed (e.g., the clock speed that a traditional Virtex routing architecture can achieve) to achieve more savings in power dissipation.

We also explored the potential benefits of increasing the percentage of HARPs inside the switch boxes by allowing a small percentage (10%) of HARPs to connect to each other. This is illustrated in Figure 11, which shows HARP resources (HARP SW) and regular switches (FlexSW) lumped to form distinct regions inside the switch boxes. This representation is just for illustration purposes. In reality, HARPs are distributed throughout the switch boxes, and not just at lower tracks.

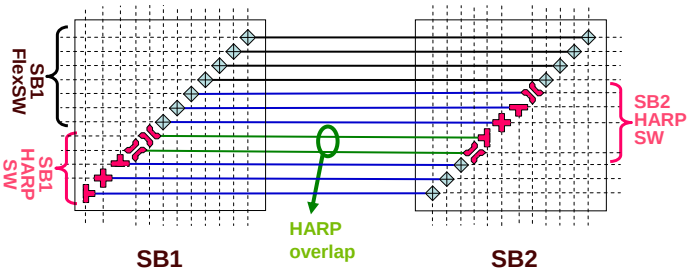


Fig. 11: Overlaps in HARPs

Allowing a small percentage of HARPs to connect directly could create more complex routing patterns to be formed by combining hardwired patterns that we have used. However, doing so could also have the undesired affect of creating dangling wire segments (as illustrated in Figure 10) which could have an adverse effect on delay and power.

In terms of implementation, to increase the percentage of HARPS beyond 50%, we relax the constraint of not allowing different HARPs to connect together and allow HARPs to connect together sometimes (*e.g.*, to allow 60% of the switches to be HARP, we should allow HARPs to connect 10% of the time). As before, we take care not to form large trees of HARPs. This is done by making sure that there is at least one regular switch after every K switches, K being a constant (in our experiments, we used $K = 3$). The results of increasing the percentage of HARPs is presented in Table II. We conducted some experiments to study the impact of increasing the overlap between HARPs beyond 60%. We observed that the results worsen as the percentage of overlaps increases beyond 70%. We believe that for each circuit, there is a sweet spot between 60% and 70% that would be optimum in terms of performance. However, it is to be noticed that going from 50% to 60% of HARPs increases the channel width by about 7% and the improvement in delay is only about 3%. This indicates that it is not advisable to go much higher than 60% as the improvement in performance comes at a cost of a much higher increase in

Circuit	Delay ($x10^{-8}$)		Area ($x10^{6}$)		Channel Width		Leakage Power		Total power		Energy ($x10^{-8}$)	
	Vtx	HRP	Vtx	HRP	Vtx	HRP	Vtx	HRP	Vtx	HRP	Vtx	HRP
misex3	6.31	5.33	2.88	2.71	20	23	0.12	0.09	0.22	0.22	1.41	1.18
alu4	7.12	5.97	3.11	2.74	19	21	0.13	0.10	0.23	0.22	1.67	1.34
apex4	7.17	5.98	2.83	2.58	22	24	0.12	0.09	0.18	0.17	1.31	0.99
ex5p	6.40	5.50	2.36	2.23	22	25	0.10	0.08	0.17	0.16	1.11	0.90
des	7.89	6.00	6.02	5.75	16	18	0.25	0.21	0.41	0.41	3.27	2.45
seq	6.39	5.31	3.58	3.46	20	24	0.15	0.12	0.27	0.27	1.73	1.43
apex2	7.45	5.92	4.01	3.80	21	24	0.17	0.13	0.28	0.28	2.09	1.66
spla	12.40	8.93	9.90	9.70	28	33	0.43	0.34	0.52	0.48	6.51	4.28
pdc	14.20	10.70	14.80	13.70	33	39	0.64	0.50	0.75	0.65	10.65	6.98
ex1010	15.50	11.50	8.95	8.66	19	23	0.38	0.31	0.48	0.45	7.39	5.15
clma	18.6	14.2	19.6	18.6	24.2	28.8	0.83	0.60	1.10	1.02	20.6	14.6
dsip	4.39	3.88	3.92	3.96	13.4	16.8	0.15	0.10	0.39	0.42	1.74	1.63
diffeq	6.08	5.15	2.22	2.18	14.8	17.6	0.09	0.07	0.18	0.17	1.08	0.910
elliptic	8.87	6.58	7.38	6.98	20	23.4	0.32	0.22	0.51	0.51	4.55	3.38
frisc	9.28	7.97	8.26	7.70	23.6	27.6	0.36	0.24	0.46	0.41	4.32	3.27
s298	10.8	9.17	3.20	2.94	17.2	18.4	0.13	0.10	0.22	0.20	2.47	1.91
s38417	8.53	7.69	10.2	9.74	16.6	18.4	0.36	0.36	0.77	0.73	6.60	5.61
tseng	5.77	5.30	1.49	1.45	13.3	16.7	0.05	0.04	0.13	0.13	0.76	0.72
bigkey	4.46	3.94	4.02	3.96	14	17.3	0.14	0.12	0.37	0.39	1.69	1.57
s38584.1	8.62	7.53	10.4	9.96	17	19	0.38	0.30	0.70	0.63	5.36	4.51
oc_wb _dma	8.86	7.41	25.1	23.8	28	33	0.99	0.77	1.15	1.15	10.24	8.50
oc_mem _ctrl	8.28	7.02	16.46	15.47	19	23	0.62	0.49	0.89	0.83	7.33	5.82
oc_des_ des3perf	14.91	12.60	56.97	54.19	16	19	2.17	1.71	2.67	2.60	39.74	32.77
G.Mean ratio (%)		82.55		95.04		116.66		77.89		96.06		79.46

TABLE I: Comparison of 50% HARPs with no HARPs. (**G.Mean** is the geometric mean)

Circuit	Delay ($\times 10^{-8}$)	Area ($\times 10^{6}$)	Channel Width	Leakage Power	Total Power	Energy ($\times 10^{-8}$)
misex3	4.98	2.68	24	0.093	0.226	1.125
alu4	5.67	2.73	22	0.0101	0.228	1.292
apex4	5.74	2.59	26	0.093	0.168	0.964
ex5p	5.6	2.19	26	0.062	0.157	0.879
des	6.36	5.66	18	0.188	0.388	2.467
seq	5.30	3.45	26	0.121	0.268	1.420
apex2	6.10	3.74	25	0.114	0.270	1.647
spla	8.31	9.38	33	0.314	0.465	3.864
pdc	9.42	13.5	40	0.490	0.645	6.075
ex1010	11.4	8.45	24	0.299	0.429	4.891
clma	12.0	19.8	32.8	0.688	1.013	12.2
dsip	3.49	3.93	18	0.129	0.401	1.40
diffeq	4.80	2.15	18.4	0.0685	0.173	0.833
elliptic	6.71	7.24	27.4	0.247	0.504	3.38
frisc	7.95	8.05	31.4	0.267	0.411	3.27
s298	8.98	2.98	20.2	0.106	0.207	1.87
s38417	7.57	9.57	19.6	0.310	0.665	5.04
tseng	5.15	1.40	17.3	0.041	0.130	0.670
bigkey	3.85	3.90	18.0	0.107	0.358	1.38
s38584.1	7.45	9.90	20	0.252	0.57	4.25
oc_wb_dma	7.19	23.80	35	0.737	1.137	8.16
oc_mem_ctrl	6.68	14.54	23	0.436	0.801	5.35
oc_des_des3perf	12.05	60.24	22	1.855	2.461	29.66
Geometric Mean	0.796	0.949	1.241	0.690	0.938	0.745

TABLE II: Comparison of 60% HARPs with no HARPs

the channel width.

We observe that increasing the percentage of HARPs(60%) inside switch boxes increases the potential savings in circuit delay, energy and area to about 21%, 26% and 5% respectively.

B. HARP Usage Analysis

To better understand the benefits and potential future improvements on the proposed HARP architecture, we perform detailed analysis on the routing results of the HARP platform. As no major changes were made in our placement and routing algorithms compared to the traditional architectures, it is of our interest to to find out how effectively the HARP connections are utilized in the final routing results. In order to find the utilization of HARP resources, we read the routing result files of the HARP architecture into the VPR Pattern Finder and compare the actual switch usage to the number of HARP resources provided in the architecture.

When comparing HARP resource availability to HARP resource utilization, we pay special attention to cases where more complex HARP connections such as the $\top$-shaped and $+$-shaped connections are only partially utilized, i.e., used as L's, H's and V's. As a result, utilization analysis, we report how many legs (e.g., up to 3 for the $\top$-shaped HARP connections and up to 4 for the $+$ connections) are actually used. Zero-leg utilization means that the switch was not used at all.

The analysis is carried on the same 20 benchmarks. For each benchmark and for each HARP connection pattern, we report the percentages for each case when a different number of legs are used (ranging from 1 to 4). Then these numbers are normalized across all the benchmarks. The final result is aggregated in Figure 12(a). The graph shows that about 79% of the overall $+$ shaped HARP resources are used in the final routing graphs. Among them, about 2.7% are fully used, i.e. all 4 hard-wired legs are involved in the connection, 10.5% use 3 hard-wired connections, and 66% use 2 hard-wired connections. The remaining 21% of this type of switch are not used in the final routing graph at all.

Based on the results shown in Figure 12(a), we can observe that all types of HARP connections are heavily used in the final routing graph. The usage of every switch type is always at least 29%. For $\top$-shaped and $+$ shaped connections, more then 50% are utilized. However, for different $\top$-shaped and $+$-shaped connections, only a very small percentage are fully utilized. For most of them, only two legs are involved in the final routing. This may indicate that they are actually used as L, $|$ or $-$ shaped connections. Moreover, considering that only small number of switches are configured as $\top$ and $+$ (refer to Figure 5(b)), this motivates the idea to simplify the HARP pattern set by getting rid of the $\top$ and $+$ patterns completely and at the meantime, providing more L, $|$ and $-$ HARP connections.

Based on the observation made above, we are interested in investigating the effect of simplifying the HARP pattern set, namely by eliminating the $\top$-shaped and $+$-shaped HARPs and redistributing their percentages to the other HARP connections. We re-evaluated the performance gain on the benchmarks with this simplified HARP pattern set. The results are presented in Table III. For the simplified HARP pattern set, we observe that the delay improves by 21%, which is about 4% better than the improvement obtained with the full pattern set. There are no significant differences in the other metrics.

Circuit	Delay ($\times 10^{-8}$)	CW	Area ($\times 10^{6}$)	Energy ($\times 10^{-8}$)
misex3	5.17	25	2.81	1.15
alu4	5.93	22.8	2.90	1.32
apex4	6.06	27.8	2.77	0.987
ex5p	5.35	28	2.36	0.877
des	6.40	18.6	5.82	2.40
seq	5.22	27	3.62	1.39
apex2	6.16	27.6	3.99	1.65
spla	8.88	35.8	9.91	4.02
pdc	9.52	43.4	1.48	6.20
ex1010	11.6	27.4	9.38	5.05
tseng	4.70	17	1.58	7.25
bigkey	3.59	17.7	4.00	1.53
s38584.1	7.12	19.3	9.96	4.66
clma	11.9	32.4	20.2	12.9
dsip	3.51	17.8	4.04	1.42
diffeq	4.80	18.4	2.25	0.863
frisc	7.74	29.8	7.95	3.15
s298	8.52	19.2	3.00	1.86
s38417	8.22	19.2	9.92	5.37
elliptic	7.39	24.8	7.02	3.36
oc_wb_dma	6.80	34	24.80	8.31
oc_mem_ctrl	7.20	23	15.30	5.92
oc_des3_des3perf	12.10	19	55.20	30.98
Geometric Mean	0.796	1.25	0.99	0.763

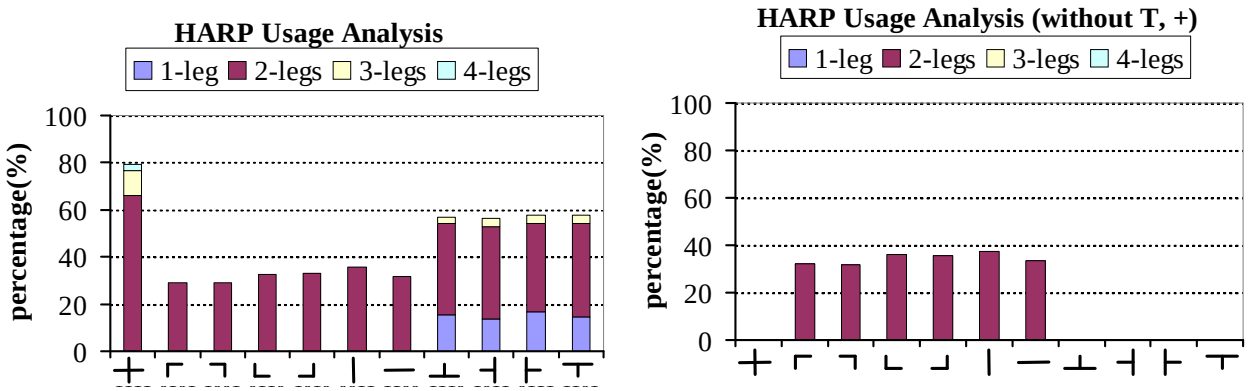
TABLE III: Results for Simplified Pattern set

The same HARP usage analysis is also performed on the routing results with the simplified HARP pattern set and the results are shown in Figure 12(b). Compared to Figure 12(a), there are no significant changes in the usage over different HARP connections.

Besides the HARP usage analysis, we are also interested in finding out whether the final routing connection behavior is changed after introducing the HARP architecture. More specifically, we want to cross check if HARP has any effect on the connection pattern distribution compared to traditional architectures.

Figure 13 extends Figure 5(b) by adding the data obtained from the routing results on the HARP architectures. It seems that it is safe to say that HARP does not change the pattern distribution characteristics fundamentally. However, one interesting observation can be made on the $|$ and $-$ patterns. With HARPs (either the full set or the simplified set) the percentage of $|$ and $-$ connections made in the final routing graphs drops significantly (especially when compared to the timing-driven results without HARPs). In the meantime, this drop seems to be compensated mainly in the increases on different L-shaped patterns. One explanation could be that by using HARP we reduce the delay of the L-patterns in the routing structure, and as a result, using L patterns becomes less costly than using an H followed by a V. Hence, the router is encouraged to use more turns in the routing paths compared to the non-HARP architectures.

In the above discussion, results in HARP usage are normalized by the number of HARP resources, while results in



(a) Original HARP set

(b) Simplified HARP set: without $\top$ and $+$

Fig. 12: HARP connection usage

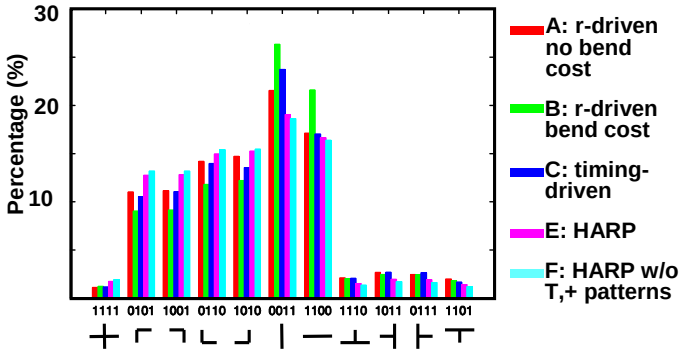


Fig. 13: Switch box pattern distribution comparison

connection pattern distribution analysis are normalized to the total number of switches utilized in the routing results. It is also interesting to see how effective the switch resources are utilized compared to the total number of switches provided by the architecture.

Table IV shows the overall switch resource utilization data. In the table, *sim-harp* refers to the simplified HARP architecture (i.e. without $\top$ and $+$ shaped HARPs), and the usage percentages reported here are computed as:

$$percentage = \frac{\text{total \# of switches used in final routing}}{\text{total number of switches provided}} \quad (1)$$

Regardless of the underlying FPGA architecture, the results in Table IV show that the circuits make poor utilization of the switch resources available in the FPGA. Moreover, without HARP-specific improvements on the placement and routing tools, this efficacy suffers a 7.8% $\sim$ 14.4% drop on the HARP architectures compared to a regular Virtex-II architecture.

VI. Tile-Based Design of HARP FPGAs

One of the most complicated tasks in the design of FPGAs is the layout. Typically, manufacturers manually layout a single tile consisting of a logic block and switch block and replicate them across the entire chip. However, when HARPs are inserted randomly in the switch-boxes, they are no longer identical and a tile based layout is not possible. In this section, we present a method that facilitates layout in the presence of HARPs.

In a multi-segment architecture, the start points of different segments are staggered to enhance routability of the architecture. Figure 14 shows a staggered arrangement of tracks with each channel having three segments of length 3[3].

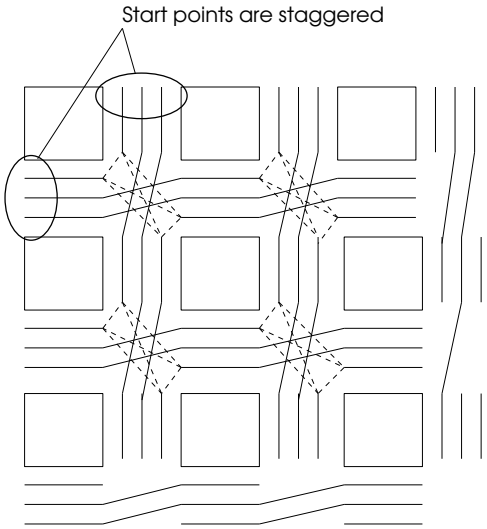


Fig. 14: Staggered Arrangement of Tracks

When a disjoint switch-box topology with a staggered arrangement of segments, a tile based layout is possible when the number of tracks of a particular length is divisible by the length of the tracks. However, layout of a single tile with HARPs in the switch-box is not possible for the following reasons:

- When all the patterns are inserted in a single switch-box to enable a tile based layout, long patterns will be formed throughout the chip. This will increase the capacitive load on the segments. The track count to route circuits will also go up since, tracks having HARP resources will

Circuit	x-size	y-size	Channel Width			Switch resource usage(%)		
			no-harps	harps	sim-harps	no-harps	harps	sim-harps
alu4	40	40	18	26	25	24.99	21.32	22.78
apex2	44	44	20	32	32	30.53	21.85	22.78
apex4	36	36	21	31	33	26.9	20.45	19.89
bigkey	54	54	18	18	18	14.71	16.22	17.03
clma	92	92	24	39	32	26.41	20.73	24.84
des	63	63	16	22	22	16.26	16.01	16.98
diffeq	39	39	14	20	21	22.88	19.74	18.72
dsip	54	54	14	18	18	15.84	16.31	15.66
elliptic	61	61	20	30	31	22.43	18.54	17.91
ex1010	68	68	22	32	24	24.87	20.59	26.75
ex5p	33	33	22	32	32	26.91	22.58	24.25
frisc	60	60	25	34	33	22.71	20.95	22.85
misex3	38	38	20	28	25	28.15	22.07	26.16
pdc	68	68	33	50	43	25.67	21.61	23.3
s298	44	44	18	23	22	23.46	21.19	22.58
s38417	81	81	17	23	23	18.53	16.83	17.65
s38584	81	81	16	18	19	17.91	17.58	18.22
seq	42	42	21	32	27	28.34	22.32	24.97
spla	61	61	28	43	36	24.76	19.88	23.86
tseng	33	33	14	16	17	20.94	19.51	19.72
Average			20.05	28.35	26.65	23.16	19.81	21.35

TABLE IV: How effective the switch resources are used in different architectures

run the entire length of the chip and only one net will be able to use the track.

- There may not be enough tracks in the channel to accurately capture the distribution of various patterns. Some of the patterns have low percentages and may not appear in the switch-box.

In order to prevent long patterns from being formed in the chip, we need to look at the neighbors of every switch-point in a switch box and ensure that they do not form such patterns. This is illustrated in figure 15. For example, when we are looking at switch point a, we also need to consider the switch points b (top) and c (right) before deciding on the type of the switch (flexible or HARP) that can be used at a. It is to be noted that we do not have to consider the other neighbors(left and bottom) of a as they would also correspond to b and c.

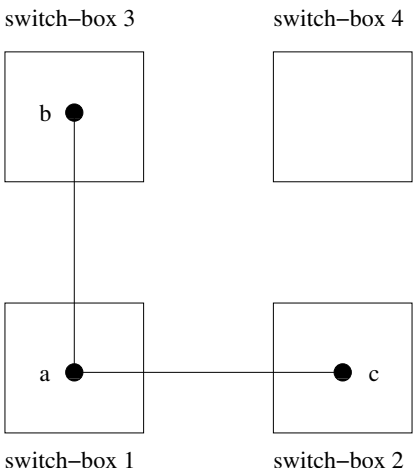


Fig. 15: Switch point neighbors

This problem can be conceptually visualized as distributing HARPs in a 2×2 switch-box array so that long patterns are not formed. We solve this problem by formulating it as an Integer programming problem with constraints to prevent forming long patterns and to ensure that the distribution follows the statistical estimation presented in section III. The rest of the section presents the formulation of the ILP.

If $ntracks$ is the number of tracks that require end-point connections at every switch-box, we have a total of $4 \times ntracks$ candidate switch-points in the 2×2 switch-box array. Each of them can be either be a flexible switch or a HARP resource based on the constraints. The HARP resources are labeled as shown in figure 16. We have not included the $+$ pattern in this study since the percentage of $+$'s is very small when compared to the others.

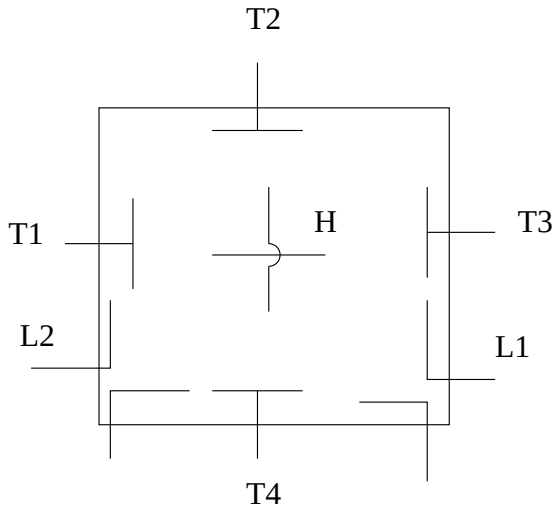


Fig. 16: HARP labeling

The following variables in the ILP are used to represent the switches:

$x_{ij}n$: flexible,

$x_{ij}h$: horizontal and vertical HARP,

$x_{ij}l_k$ k=1,2 : L_1, L_2

$x_{ij}t_k$ $k = 1 \cdots 4$: T_1, T_2, T_3, T_4

Where i is the switch-box number and j is the track num-

ber. To maximize routing flexibility, we should ensure that flexible switches are distributed evenly between switch-boxes. To this end, we try to reduce the maximum number of flexible switches that are inserted in each switch-box. This coupled with the capacity constraints on the number of flexible switches and HARP resources will produce the right mixture of flexible and HARP resources in all switch-boxes. Thus the objective function is written as follows:

$$Minimize \quad W$$

subject to,

$$W \geq n_i \qquad i = 1 \cdots 4 \tag{2}$$

$$n_i = \sum_{j=1}^{ntracks} x_{ij} n \quad i = 1 \cdots 4 \tag{3}$$

Each of the x_{ij} variables can be either 0 or 1 depending on the type of switch present at switch box i and track j. Writing this down in the form of a constraint,

$$x_{ij}n, x_{ij}l_k, x_{ij}t_k, x_{ij}h \in \{0, 1\} \tag{4}$$

At every switch-point, we can have only one kind of switch. This means that exactly one of the x_{ij} variables is 1 and the rest are 0. This gives rise to the following constraint:

$$x_{ij}n + x_{ij}h + \sum_{p=1}^{2} x_{ij}l_p + \sum_{q=1}^{4} x_{ij}t_q = 1, \tag{5}$$

To prevent forming long horizontal and vertical patterns that run throughout the chip, restrictions are imposed on the location of HARPs inside different switch-boxes. For example, if two horizontal or vertical HARPs were to connect to each other and the patterns are replicated, it would form a long pattern throughout the chip. To avoid this, we have a constraint that prevents horizontal or vertical HARPs in adjacent switch locations. Similar to the example mentioned, there are certain restrictions on the locations of other patterns as well depending on their orientations. These are captured by the following set of constraints:

$$x_{ij}h + x_{targetj}h \leq 1, \quad \forall \quad i, j \tag{6}$$

$$\begin{aligned} if\ i = 4 \quad target &= 1, \\ else\ target &= i + 1 \end{aligned}$$

$$\begin{aligned} x_{pj}t_p + x_{qj}t_q &\leq 1 \\ x_{pj}t_p + x_{qj}h &\leq 1 \\ x_{pj}h + x_{qj}t_p &\leq 1 \end{aligned} \tag{7}$$

$$(p, q) \in \{(1, 2), (2, 3), (3, 4), (4, 1)\}$$

To prevent forming loops with HARPs, we impose a constraint that for any HARP resource, at least one of its two neighbors is a flexible switch. This can be written as

$$x_{pj}n + x_{qj}n + x_{rj}n \geq 1 \tag{8}$$

$$(p, q, r) \in \{(1, 2, 4), (2, 1, 3), (2, 3, 4), (3, 4, 1)\}$$

Finally, we need to add capacity constraints on the number of various patterns that need to be present based on the required distribution.

$$\begin{aligned} \sum_{i=1}^{4} \sum_{j=1}^{ntracks} x_{ij}h &= H_{required} \\ \sum_{i=1}^{4} \sum_{j=1}^{ntracks} x_{ij}l_k &= L_{k\ required} \quad k = 1, 2 \\ \sum_{i=1}^{4} \sum_{j=1}^{ntracks} x_{ij}t_k &= T_{k\ required} \quad k = 1, \cdots 4 \end{aligned} \tag{9}$$

The solution of the above ILP problem gives the location of the HARPs in the switch-boxes. It is to be noted that for segments that span C logic blocks, the neighbors(x_{ij}'s) correspond to switch-boxes that are spaced C units apart. This information is used to design a limited number of switch-boxes which are then replicated throughout the chip.

Circuit	Delay ($\times 10^{-8}$)	Area ($\times 10^{6}$)	CW	Energy ($\times 10^{-8}$)
misex3	4.90	3.15	28	1.2
alu4	5.91	3.13	24	1.41
apex4	6.04	2.62	26	1.03
ex5p	6.52	2.68	29	1.00
des	5.95	6.32	20	2.41
seq	5.27	4.22	30	1.54
apex2	5.48	4.03	30	1.86
spla	8.06	11.7	43	4.32
pdc	1.25	16.1	45	8.25
ex1010	1.35	10.4	29	6.63
oc_wb_dma	7.45	24.9	34	8.88
oc_mem_ctrl	7.91	16.18	24	6.425
oc_des_des3perf	12.78	60.95	21	31.55
Geometric Mean	**0.828**	**1.06**	**1.35**	**0.82**

TABLE V: Results of ILP formulation

A. Results

We placed and routed benchmark circuits on the architecture generated using the solution to the ILP formulation and present the results in Table V. We observe that when HARPs are distributed in switch-boxes and replicated througout the chip, the performance worsens when compared with the case when they are distributed randomly.

When compared with the traditional architecture, delay improves by the same amount as that obtained with random distribution of HARPs inside switch-boxes. However, we observe an 6% increase in the area. This is attributed to an increase in channel width. Even though we avoid forming long patterns inside switch-boxes, distributing HARPs inside a few switch-boxes and replicating them restricts the freedom of the router. When patterns are distrbuted randomly, the router has different options at every switch-box and this provides more flexibility. With random distribution, we observed a slight reduction in the area inspite of the increase in channel width since the average area of each switch-box reduced due to HARPs. In this case however, the increase in channel width offsets this and a net increase in area is observed. Though the area increases by about 6%, area-delay product which is

a useful metric for comparing different architecures reduces by about 13% and the overall energy consumption reduces by 18%. This coupled with the fact that switch-box layout is not an issue anymore makes HARP an attractive design choice.

The results we have reported in Table V are probably a bit optimistic. Since the area increases, the routing segments would have to travel longer distances and their increased parasitics would have a negative impact on the signal delays that are mapped to them. Due to our limited resources we did not attempt laying out the HARP FPGA and hence cannot report accurate delay and area numbers in this paper.

VII. Discussion and Conclusion

We propose a technique to reduce circuit delay, area and power dissipation by introducing hard-wired patterns inside switch boxes. The population of the HARPs is guided by statistical analysis of routing trees that are generated on a traditional architecture by the VPR tool. We analyzed the routing profiles of various circuit benchmarks and came up with a statistical measure of the routing patterns present inside the switch boxes. The routing graph construction of VPR was modified to include these patterns. Simulation results after detailed routing showed a potential improvement of 20% in circuit delay, 5% in the circuit area, 33% in the leakage power dissipation and about 6% in the total power dissipation. We observed that by introducing hardwired patterns, we can considerably speed up the circuit and at the same time achieve reasonable savings in circuit area and power dissipation.

In Section III-A we mentioned that the placement and routing algorithm and the architecture will affect the outcome of the statistical analysis. In Section III-B.2 we showed that the architecture has a bigger role compared to the routing algorithm. But nevertheless, both the physical design algorithms and the architecture will skew the pattern frequency analysis. Apart from the fact that there is a certain degree of inevitability in this influence, we argue that such effect could be considered useful. We would like to generate the architecture with an eye on the CAD algorithms (*e.g.*would the results change if we use an alignment based placement method such as [15]?). If we create an architecture that conforms to the behavior of the placement and routing algorithms, the potential benefits will be greater.

Further work is needed in making the placer aware of the changes in the routing architecture. We also need to look at the possibility of modifying Steiner tree routing algorithms to make full use of the hard-wired patterns and to achieve better correlation between the placement tool, routing tool and the routing architecture.

References

[1] J. H. Anderson, F. Najm, and T. Tuan. "Active Leakage Power Optimization for FPGAs". In Proc. of ACM/SIGDA International Symposium on Field programmable gate arrays, 2004.

[2] V. Betz and J. Rose. "VPR: A New Packing, Placement and Routing Tool for FPGA Research". In *International Workshop on Field-programmable Logic and Applications*, 1997.

[3] V. Betz, J. Rose, A. Marquardt. Architecture and CAD for Deep-Submicron FPGAs. Kluwer Academic Publishers, 1999.

[4] Y.-W. Chang, Y.-T. Chang, "An Architecture-Driven Metric for Simultaneous Placement and Global Routing for FPGAs," In Proc. of ACM/IEEE Design Automation Conference, 2000, pp. 567-572.

[5] Y.-W. Chang, D. F. Wong, "Universal Switch Modules for FPGA Design," ACM Trans. Design Automation of Electronic Systems, Vol. 1, No. 1, Jan. 1996, pp. 80-101.

[6] A. DeHon, "Reconfigurable architectures for general-purpose computing", Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.

[7] A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, and T. Tuan "A Dual-V_{DD} Low Power FPGA Architecture". In *Proceedings of International Conference on Field Programmable Logic and Applications*, 2004.

[8] HARP Architecture Generator and P&R Tool URL, `http://www.ece.umn.edu/users/kia/` (click on the Download link)

[9] The International Techonlogy Road Map for Semiconductors, 2003 Edition.

[10] M. Imran Masud. FPGA Routing Structures: A Novel Switch Block and Depopulated interconnect Matrix Architecture. M.A.Sc. Thesis, University of British-Columbia, 1999.

[11] N. Jayakumar and S. P. Khatri, "A METAL and VIA Maskset Programmable VLSI Design Methodology using PLAs", In *Proceedings of International Conference on Computer Aided Design*, 2004.

[12] R. Kastner, E. Bozorgzadeh,and M. Sarrafzadeh, "Predictable Routing", In *Proceedings of International Conference on Computer Aided Design*, 2000.

[13] R. Kastner, E. Bozorgzadeh,and M. Sarrafzadeh, "Pattern routing: use and theory for increasing predictability and avoiding coupling", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 7, No. 7, Pages 777-790, 2002.

[14] M. Khellah, S. Brown, Z. Vranesic, "Minimizing Interconnection Delays in Array-based FPGAs," In Proc. of IEEE Custom Integrated Circuits Conference, 1994, pp. 181-184.

[15] P. Maidee, C. Ababei, and K. Bazargan, "Fast Timing-driven Partitioning-based Placement for Island Style FPGAs", In *Proc. ACM/IEEE Design Automation Conference (DAC)*, 2003, pp. 598-603

[16] K. Poon, A. Yan, and S. Wilton, "A flexible Power Model for FPGAs". In *Proceedings of International Conference on Field Programmable Logic and Applications*, 2002.

[17] J. Rose, S. Brown, "Flexibility of interconnection structures for field-programmable gate arrays," IEEE J. Solid-State Circuits, 26, 3, 1991, pp. 277-282.

[18] K.Y. Tong, V. Kheterpal, V. Rovner, L. Pileggi, H. Schmit "Regular Logic Fabrics for a Via Patterned Gate Array (VPGA)". In *Proceedings of IEEE Custom Integrated Circuits Conference*, 2003.

[19] VPR Pattern Finder URL, `http://www.ece.ucsb.edu/~express/software.html`, 2004

[20] S. Wilton. Architecture and Algorithms for Field Programmable Gate Arrays with Embedded Memory. Ph.D. Thesis, University of Toronto, 1997.

[21] XC4000 FPGA Family Data Sheet. Xilinx, Inc.

[22] Y. Ran, and M. Marek-Sadowska, Designing a Via-Configurable Regular Fabric. In Proceedings of Custom Integrated Circuits Conference (2004).

[23] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0", *Tech. Report, Microelectronics Centre of North carolina*, 1991.

[24] B. Zahiri, Structured ASICs: Opportunities and challenges. In *Proceedings of International Conference on Computer Design (2003), pp. 404409.*

[25] P. S. Zuchowski, C. B. Reynolds, R. J. Grupp, S. G. Davis, B. Cremen and B. Troxel, "Hybrid ASIC and FPGA Architecture", *International Conference on Computer-Aided Design (ICCAD)*, pp. 187 – 194, 2002.

[26] Andy G. Ye and Jonathan Rose, "Using bus-based connections to improve field-programmable gate array density for implementing datapath circuits", In Proc. of ACM/SIGDA International Symposium on Field programmable gate arrays, pp. 3-13, 2005.

[27] Satish Sivaswamy, Gang Wang, Cristinel Ababei, Kia Bazargan, Ryan Kastner, and Eli Bozorgzadeh, "HARP: Hardwired Routing Pattern FPGAs", in International Symposium on Field Programmable Gate Arrays (FPGA), pp. 21 - 29, 2005.

Gang Wang (M98) was born in Shaanxi, China. He received his Bachelor of Electrical Engineering degree from Xian Jiaotong University in 1992, and Master of Computer Science degree from Chinese Academy of Sciences in 1995, both in China. From 1995 to 1997, he conducted research work in Pattern Recognition and Image Processing Lab at Michigan State University (East Lansing, MI, US), and Interactive System Lab at Carnegie Mellon University (Pittsburgh, PA, US), focusing on speech and image recognition. In 1997, he joined Computer Motion Inc. and worked as a Principle Engineer on research and development of surgical robotics systems. Currently, he is a PhD student at the Department of Electrical and Computer Engineering, University of California at Santa Barbara. His research interests include evolutionary computation, reconfigurable computing, computer-aided design and design automation.

Satish Sivaswamy received his B.E degree in Electrical and Electronics Engineering from the University of Madras, India in 2002. Presently he is working towards a PhD degree at the University of Minnesota. His research interests are in physical design for FPGAs, FPGA architectures and reconfigurable computing.

Cristinel Ababei received the Ph.D. degree in electrical engineering from the University of Minnesota in 2004 and the B.S. degree in microelectronics from the Technical University of Iasi, Romania, in 1996. He joined Magma Design Automation in 2004. His research interests include CAD for layout and logic synthesis for robust VLSI circuits and FPGAs.

Kia Bazargan received his Bachelors degree in Computer Science from Sharif University in Tehran, Iran, and his M.S. and PhD in Electrical and Computer Engineering from Northwestern University in Evanston, IL in 1998 and 2000 respectively. He is currently an Assistant Professor in the Electrical and Computer Engineering at the University of Minnesota. He has served on the technical program committee of a number of IEEE sponsored conferences (e.g., ISPD, ICCAD, ASPDAC, GLSVLSI). He was a guest co-editor of ACM Transactions on Embedded Computing Systems (ACM TECS), Special Issue on Dynamically Adaptable Embedded Systems in 2003. He is an Associate Editor of IEEE Transaction on Computer- Aided Design of Integrated Circuits and Systems. He was a recipient of NSF CAREER award in 2004. His research interests are computer-aided design, FPGAs and reconfigurable computing.

Ryan Kastner is an assistant professor in the Department of Electrical and Computer Engineering at the University of California, Santa Barbara. He received a PhD in Computer Science (2002) at UCLA, a masters degree in engineering (2000) and bachelor degrees (BS) in both electrical engineering and computer engineering (1999), all from Northwestern University. His current research interests lie in the realm of embedded system design, in particular reconfigurable computing, wireless communications and underwater sensor networking.

Professor Kastner has published over 50 technical articles, and is the author of the book, "Synthesis Techniques and Optimizations for Reconfigurable Systems", available from Kluwer Academic Publishing. He is a member of numerous conference technical committees including International Conference on Computer Aided Design (ICCAD), Design Automation Conference (DAC), International Conference on Computer Design (ICCD), Great Lakes Symposium on VLSI (GLSVLSI), the the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA) and the International Symposium on Circuits and Systems (ISCAS). He serves on the editorial board for the Journal of Embedded Computing.

Elaheh Bozorgzadeh (S99M03) received the B.S. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 1998, the M.S. degree in computer engineering from Northwestern University, Evanston, IL, in 2000, and the Ph.D. degree in computer science from the University of California, Los Angeles, in 2003. She is currently an Assistant Professor in the Department of Computer Science, University of California, Irvine. Her research interests include very large scale integrated computer-aided design, design automation for embedded systems, and reconfigurable computing. Prof. Bozorgzadeh is also a member of the ACM.