HARP: Hard-wired Routing Pattern FPGAs

Satish Sivaswamy[†], Gang Wang[‡], Cristinel Ababei[†],
Kia Bazargan[†], Ryan Kastner[‡] and Eli Bozorgzadeh[††]

[†]ECE Dept.
Univ. of Minnesota
Minneapolis, MN 55455
satish,ababei,kia@ece.umn.edu

[‡]Dept. of ECE
Univ. of California, Santa Barbara
Santa Barbara, CA 93106
wanggang,kastner@ece.ucsb.edu

[††]Computer Science Dept.
Univ. of California, Irvine
Irvine, CA 92697
eli@igor.ics.uci.edu

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[1] that utilizes a mixture of hard-wired and traditional flexible switches. The result is 24% reduction in leakage power consumption, 7% smaller area and 24% shorter delays, which translates to 30% increase in clock frequency. Despite the increase in clock speeds, the overall power consumption is reduced by 8%.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Style—*Gate arrays*; B.7.2 [**Integrated Circuits**]: Design Aids—*Routing*

General Terms

Design, Performance, Experimentation

1. INTRODUCTION

Prohibitive ASIC mask costs and stringent time-to-market windows have made FPGAs an attractive implementation platforms 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:

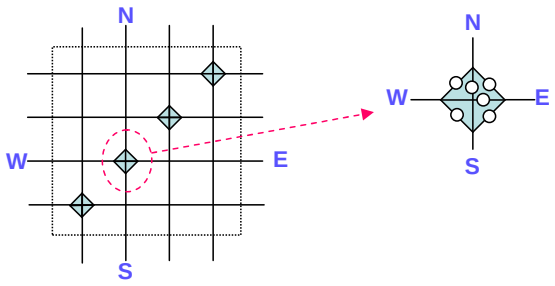


Figure 1: SRAM-based Switch Box.

- **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.
- **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].

In this work, we extend the idea of eliminating switches 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 13. 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. The distribution of the hard-wired routing patterns (HARPs) are obtained after a careful analysis of the routing profiles of different circuits. Our technique maintains the programmability of FPGAs, while improving their performance metrics. We place and route circuits with these patterns embedded in the switch boxes and report results of area, delay, power and channel width.

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

FPGA'05, February 20–22, 2005, Monterey, California, USA.

The paper is organized as follows. In Section 2, we describe the terminology that we use throughout the paper and the overall flow of our architectural design. In Section 3, 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 3.3. Section 4 explains how hard-wired routing patterns inside the switch boxes are exploited by the router. Experimental results are presented in Section 5. Section 6 gives details on related work. We conclude in Section 7, by outlining our main contribution and discussing future research directions.

2. PRELIMINARIES

2.1 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: ⊤,⊥, ⊣, ⊢, ⌜, ⌝, ⌞, ⌟, |,− and +.

2.2 HARP-based FPGA Routing Architecture Design Flow

Figure 2 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.

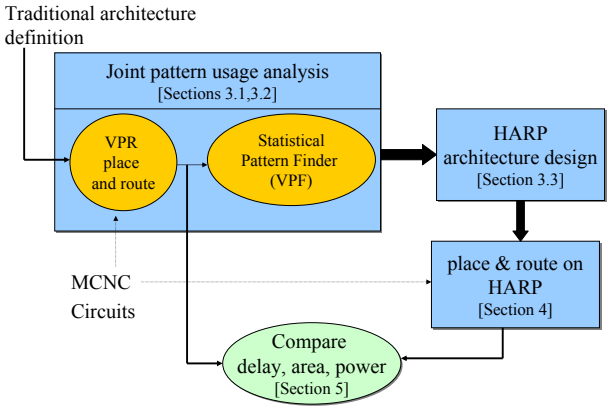


Figure 2: HARP-based Routing Architecture Design Flow.

3. ROUTING PATTERN ANALYSIS

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

3.1 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 7.

Statistical information can guide us on how often we need to insert HARP switches 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, we routed each circuit, detected the patterns in the route files, and applied statistical analysis on the data.

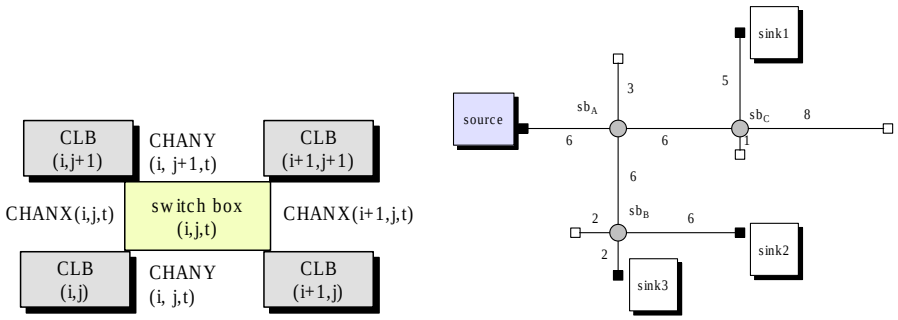
In our studies, we considered two segmentation architectures: unit-length segmentation and multi-length segmentation (similar to Xilinx's Virtex family). We used the VPR router in three different modes: Timing-driven, Routability-driven without bend-cost, and Routability-driven with bend-cost. We experimented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

3.2 Analysis of Routing Patterns

3.2.1 VPR Pattern Finder Tool

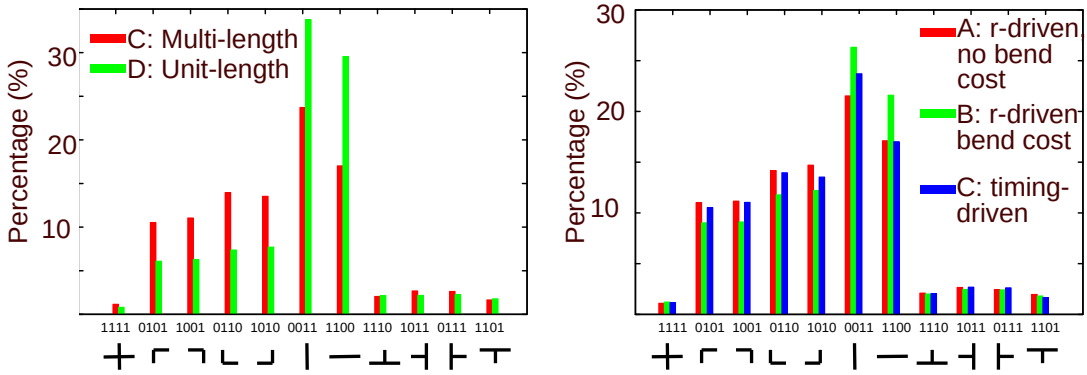
In order to analyze the behavior of the routing patterns, we have implemented *VPR Pattern Finder*(VPF), a graphic tool for parsing, visualizing and analyzing the VPR routing results [19]. VPF takes a VPR routing result file as input and automatically extracts the routing information, identifies the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

Based on the underlying FPGA architectures, there are two types of VPR routing file formats: *unit-length* or *multi-length*. In the unit-length architecture, each routing segment has a length of one, while in the latter approach, segments span multiple configurable logical block (CLBs), and are staggered to provide faster and more direct connectivity. A multi-length segment is defined by its starting and ending coordinates. In VPF we provide a uniform model to handle both formats, where the unit-length results are treated in the same way for multi-length segments, except the starting coordinates and the ending coordinates are identical.



(a) switch box indexing (b) pattern labelling

Figure 3: Switch box indexing and pattern labelling in VPF



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

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

Figure 4: Switch box pattern distributions
(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)

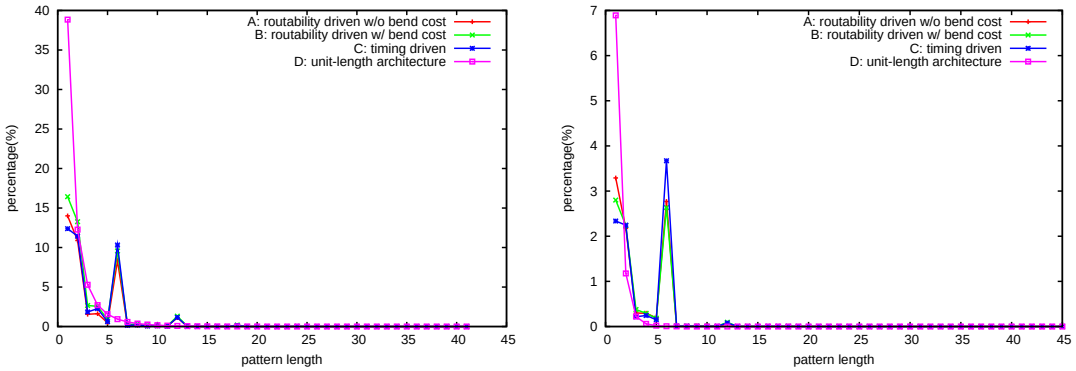
In our analysis, we focus on the connection patterns of the switch boxes. For a given FPGA layout, a switch box is indexed by a tuple (i, j, t) as shown in Figure 3(a), where i and j indicate the physical location of the switch box and t identifies the track being used. Based on the structure of the switch box, we have 11 possible connection patterns on a switch box. They are $\top$, $\bot$, $\dashv$, $\vdash$, $\ulcorner$, $\urcorner$, $\llcorner$, $\lrcorner$, $|$, $-$ and $+$. We encode these patterns with four binary bits that denote whether the *left, right, top* and *bottom* track associated with the switch box is connected to the junction. As an example, Figure 3(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 $\top$(1101), sb_B has pattern $\vdash$(0111), and sb_C is of pattern $\lrcorner$(1010).

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 box that has a pattern other than $|$ or $-$, or 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 box. Following the same example shown in Figure 3(b) and assuming the numbers by the segments indicate the segment lengths, switch boxes sb_A, sb_B and sb_C have pattern lengths 6, 2 and 5 respectively.

3.2.2 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 4 shows the normalized pattern distributions (in percentage of all the switch boxes) 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 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.

One interesting observation that can be made from Figure 4(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



(a) Length distribution for pattern | and −

(b) Length distribution for T patterns

(c) Length distribution for L patterns

(d) Length distribution for pattern +

Figure 5: Distribution statistics on switch box pattern lengths
(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)

63.3% to 41.2%. On the other hand, this drastic change is not seen in Figure 4(b) where the results are obtained on the same Virtex style architecture but with different router settings. In other words, the architecture seems to have a much bigger impact on the switch box pattern distribution than the routing algorithm. Figure 4(a) shows there is little change in the percentage of the T patterns (pattern 1110, 1011, 0111, and 1101) or the + pattern (pattern 1111) when we switch from unit-length to the multi-length segment architecture. On the contrary, there is a significant increase for the L patterns (pattern 0101, 1001, 0110 and 1010). 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 box is comparable to (if not more than) that of a vertical or horizontal pattern. This is a profound difference when compared with the unit-length results, in which the vertical and horizontal patterns are overwhelmingly dominating the pattern distribution.

Next, we analyze the length of the patterns using the method discussed in Section 3.2.1. Figure 5 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 box patterns with the given length.

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 more sophisticated 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 no surprising because in the multi-length architecture, the majority of the segments are of length 6, where switch connections are allowed at both the middle point and the 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].

3.3 Architecture Design

Architecture design for FPGAs is a complex problem and much work has been done in this area since FPGAs were 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. Among these factors, switch-block design is of paramount importance. Switch box design has been addressed in previous works.

Previously proposed switch boxes include (see Figure 6.): (i) The Xilinx XC4000-series [21] (also known as *disjoint* or *sub-*

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

set). It is area-effective but creates disjoint routing domains. (ii) *Anti-symmetric* switch box [17] is known for good practical routability. (iii) *Universal* switch box [5] can simultaneously route all two-point connections in the switch-block. (iv) *Wilton* switch box [20] eliminates the problem of routing domains and provides greater routing flexibility. This switch-block style was improved later in [10], to become *Imran* switch box as a combination of the disjoint and Wilton styles, which leads to a better trade-off between switch-block area and routability.

All of the previous works assume a switch-block flexibility $F_s = 3$, which means that every track entering and ending at one side of the switch-block will connect to three other tracks at the other three sides of the switch-block. This ensures a good practical compromise between the routing flexibility offered by the switch-block and its area as well as the run-time of the routing algorithms. Based on the analysis presented in

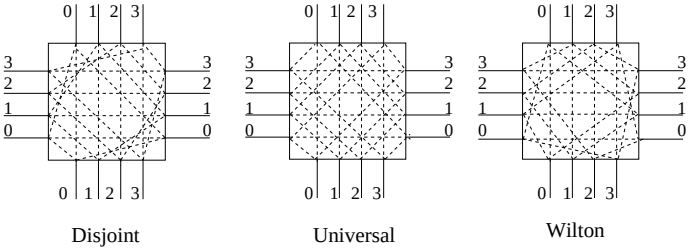


Figure 6: Typical Switch Box Configurations

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 switches on the routing of circuits (see Section 5).

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.

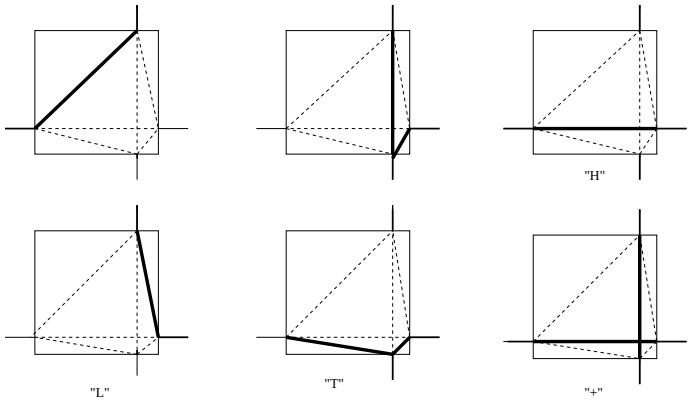


Figure 7: Some possible hard-wired patterns

4. 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 maybe 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].

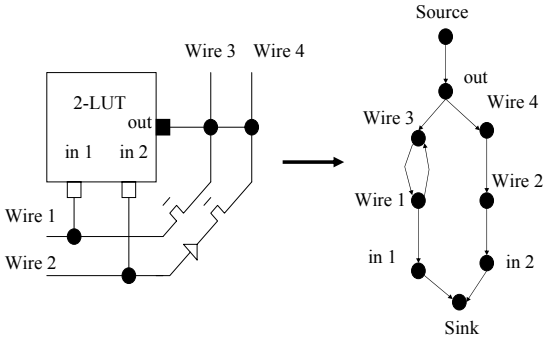


Figure 8: Sample routing graph

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 T-shaped hard wired pattern embedded in it. With the T-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.

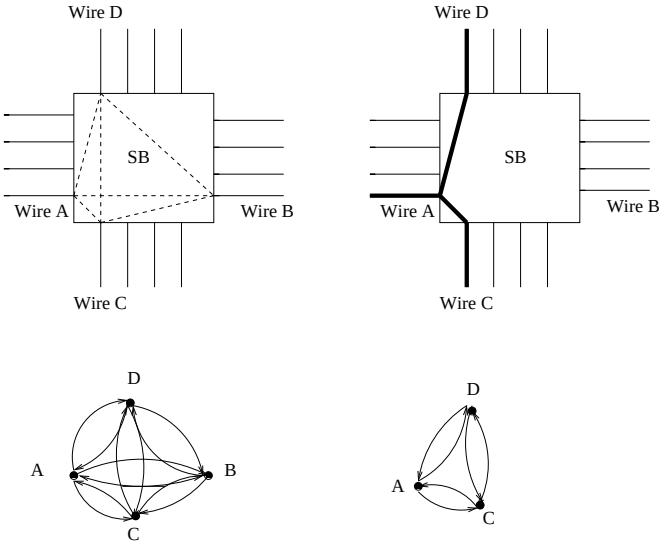


Figure 9: Routing graph with HARPs

Based on the results of the analysis presented in Section 3, 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 take care 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 5, we relax this restriction a little and observe that the *limited* use of merged HARPs will improve the quality of the circuit.

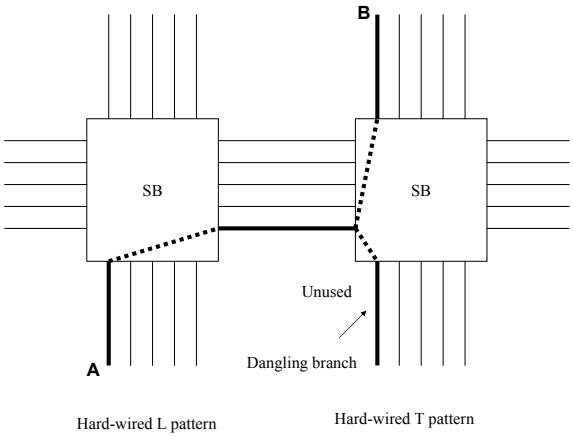


Figure 10: Connecting Hard-wired patterns together

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[3]. 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. The pseudo-code for inserting the hard-wired patterns inside switch boxes is shown in Algorithm 1.

Figure 11 shows how the pattern distribution array used in Algorithm 1 is populated. An array size of 10 is used for illustration purposes. In the figure, V, C, H, T and L denote Vertical, Cross, Horizontal, T and L shaped HARPs (no orientation considered yet). The percentages in the figure are 20%, 10%, 20%, 20% and 30% respectively and the patterns are distributed uniformly in the array (these numbers are taken from the statistical analysis of the routing pattern frequencies).

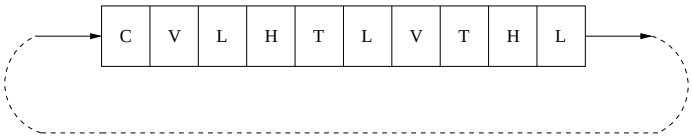


Figure 11: Initialization of Pattern Distribution Array (P)

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 13 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.

[3]For example, the "T" shapes of Figure 13 eliminate one edge from the side that T does not connect to.

Algorithm 1 Pseudo-code to Insert HARPs

Input:
Tech mapped netlist .net G(V,E)
Architecture description file
Initialization:
Initialize switch box(SB) matrix of dimensions (n_x,n_y,CW) to *unknown*
/*n_x, n_y denote number of channels in x and y directions and CW denotes channel width*/
Initialize Pattern distribution array(P) of size 100 with symbolic entries denoting various HARPs according to the frequency of distribution.
$Pattern\_Counter \leftarrow 1$
Algorithm:
for $i = 1$ to n_x **do**
  for $j = 1$ to n_y **do**
    for $track = 1$ to CW **do**
      $orientation \leftarrow$ Random {0,90,-90,180}
      $PatternID \leftarrow pattern\_id(P(pattern\_counter\%100), orientation)$
      $S \leftarrow$ switches adjacent to switch(i,j,track)
      /*Here we assume a disjoint SB with F_s=3 and segments of lengths 1,2 and 6.*/
      $patternIncompatible \leftarrow false$
      for $\forall s \in S$ **do**
        if s and $PatternID$ make two joined HARPs **then**
          $patternIncompatible \leftarrow true$
        end if
      end for
      if $patternIncompatible == false$ **then**
        $SB(i, j, track) \leftarrow PatternID$
        $pattern\_counter \leftarrow pattern\_counter + 1$
      else
        $SB(i, j, track) \leftarrow regular\_switch$
        /*This denotes a regular switch where all sides of the SB participate in the connections*/
      end if
    end for
  end for
end for

/*To insert edges in the routing graph*/
for all Switch Boxes for i in $1 \ldots n_x$, j in $1 \ldots n_y$ **do**
  for $track = 1$ to CW **do**
    if $SB(i, j, track) > 1$ **then**
      Insert only the edges forming the pattern into the routing graph
      /*For example, if a T pattern is formed, then the north segment should be eliminated.*/
    else
      Insert all possible connections into the routing graph
    end if
  end for
end for

4.1 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 switches. 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 switches, 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). This is illustrated in Figure 12.

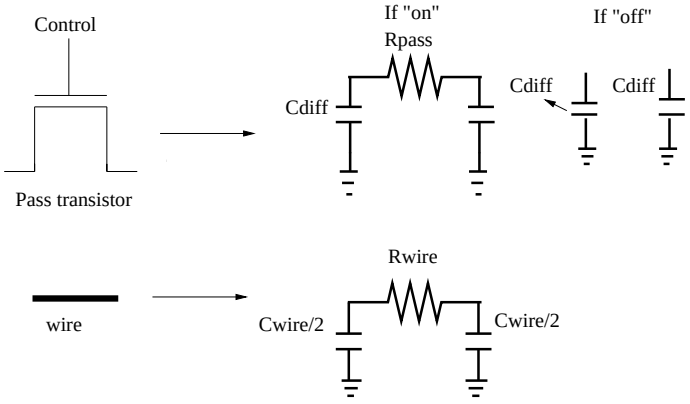


Figure 12: Modeling Pass transistor Switches

To accurately determine the delay of using a hard-wired switch, 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. The area model in VPR is based on counting the number of transistors required to implement the FPGA architecture and 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 switches, 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. Since we replaced the pass transistors with the resistance and capacitance values of metal wires for hard-wired switches, the power results reported in our paper accurately reflect the effect of these hard-wired patterns.

5. EXPERIMENTAL RESULTS

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 (similar to the Virtex architecture) for all simulation experiments. HARPs were not inserted on long lines, though. We placed and routed 10 MCNC circuit benchmarks of the VPR package on HARP architectures and report the results of circuit delay, area, leakage power, total power dissipation and channel width.

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. 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. As a result, the delay estimated at placement will not be an accurate representation of the delay at routing, and placement optimizations are not as effective. Consequently, we decided to leave the delay tables untouched from the original implementation in order to account for the fact that the router may not always be able to use HARPs for routing.

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 1[4]. Columns labeled "Vtx" show the results of the traditional "Virtex-like" routing architecture. The last two rows of the table show average values for Vtx and HARP, and the ratio between HARP average and Vtx average.

We observe that the insertion of hard-wired routing patterns has a profound impact on delay and leakage power dissipation, reducing delay by about 21.7% and leakage power by 19.7% 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 5.3% on average. However, the average channel width increases by around 15.4%. 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 6.23%. In spite of the 20% reduction in the leakage power dissipation, the total power reduces by only around 6% 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 30% 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.

Another point to be observed is that the performance of the hard-wired patterns is considerably better for larger circuits. From Table 1, we observe that for the 3 biggest circuits, spla, pdc and ex1010, introducing hard-wired patterns cause an improvement of 26% in the circuit delay and 34% in the total energy consumption. Leakage power is becoming increasingly more important compared to other sources of power consumption according to ITRS [9]. As a result, our proposed HARP architecture will become more effective as technology advances.

We also explored the potential benefits of increasing the per-

[4]Results for only 10 of the benchmarks are reported for want of space. The others yield similar results

Circuit	Delay (x10^{-8})		Area (x10^{6})		Channel Width		Leakage power		Total power		Energy (x10^{-8})	
	Vtx	HARP	Vtx	HARP	Vtx	HARP	Vtx	HARP	Vtx	HARP	Vtx	HARP
misex3	6.31	5.33	2.88	2.71	20	23	0.119	0.094	0.223	0.222	1.407	1.183
alu4	7.12	5.97	3.11	2.74	19	21	0.129	0.096	0.235	0.225	1.673	1.343
apex4	7.17	5.98	2.83	2.58	22	24	0.117	0.092	0.183	0.166	1.312	0.993
ex5p	6.40	5.50	2.36	2.23	22	25	0.098	0.080	0.173	0.164	1.107	0.902
des	7.89	6.00	6.02	5.75	16	18	0.251	0.209	0.415	0.408	3.274	2.448
seq	6.39	5.31	3.58	3.46	20	24	0.149	0.121	0.271	0.270	1.732	1.433
apex2	7.45	5.92	4.01	3.80	21	24	0.168	0.134	0.281	0.281	2.093	1.663
spla	12.40	8.93	9.90	9.70	28	33	0.428	0.342	0.525	0.479	6.51	4.278
pdc	14.20	10.70	14.80	13.70	33	39	0.637	0.505	0.750	0.652	10.65	6.976
ex1010	15.50	11.50	8.95	8.66	19	23	0.378	0.314	0.477	0.448	7.393	5.152
Avg	9.083	7.114	5.844	5.533	22	25.4	0.2474	0.1987	0.3533	0.3315	3.715	2.637
Ratio		78.32%		94.68%		115.45%		80.32%		93.83%	70.98%	

Table 1: Comparison of 50% HARPs with no HARPs

centage of HARPs inside the switch boxes by allowing a small percentage (10%) of HARPs to connect to each other. This is illustrated in Figure 13, which shows HARP switches (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.

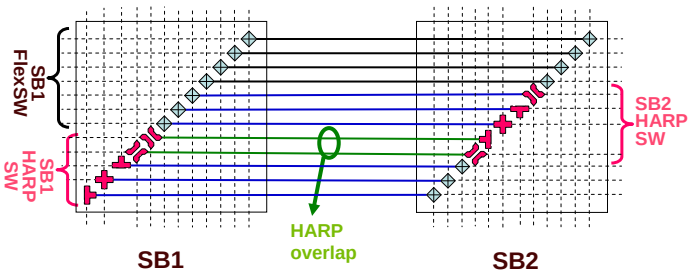


Figure 13: Overlaps in HARPs

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. Part of our future work is to find the best percentage of HARPs that could connect directly. In terms of implementation, to increase the percentage of HARPS beyond 50%, we relax the constraint present in algorithm 1 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 2.

We observe that increasing the percentage of HARPs(60%) inside switch boxes increases the potential savings in circuit delay, energy and area to about 24%, 34% and 7% respectively.

6. 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. This paper was inspired by our initial work on pattern routing [12, 13]. This work focused on the concept of using prespecified patterns to route a net. By doing so, we allow a more accurate prediction mechanism for metrics such as congestion and wirelength earlier in the design flow. The work 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.

7. 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 24% in circuit delay, 7% in the circuit area, 24% in the leakage power dissipation and about 8% 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 3.1 we mentioned that the placement and routing algorithm and the architecture will affect the outcome of the statistical analysis. In Section 3.2.2 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. If we create an architecture that con-

Circuit	Delay x(10^{-8})	Area x(10^{6})	Channel Width	Leakage power	Total power	Energy x(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
Average	6.88	5.437	26	0.188	0.324	2.463
Ratio	0.758	0.930	1.20	0.760	0.917	0.663

Table 2: Comparison of 60% HARPs with no HARPs

forms 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.

8. 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, 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.