Exploration of Pipelined FPGA Interconnect Structures

Akshay Sharma
Electrical Engineering
University of Washington
Seattle, WA
akshay@ee.washington.edu

Katherine Compton
Electrical & Computer Engineering
Northwestern University
Evanston, IL
kati@ece.northwestern.edu

Carl Ebeling
Computer Science & Engineering
University of Washington
Seattle, WA
ebeling@cs.washington.edu

Scott Hauck
Electrical Engineering
University of Washington
Seattle, WA
hauck@ee.washington.edu

ABSTRACT
In this work, we parameterize and explore the interconnect structure of *pipelined* FPGAs. Specifically, we explore the effects of interconnect register population, length of registered routing track segments, registered IO terminals of logic units, and the flexibility of the interconnect structure on the performance of a pipelined FPGA. Our experiments with the RaPiD [4] architecture identify tradeoffs that must be made while designing the interconnect structure of a pipelined FPGA. The post-exploration architecture that we found shows a 19% improvement over RaPiD, while the area overhead incurred in placing and routing benchmarks netlists on the post-exploration architecture is 18%.

Categories and Subject Descriptors
B.7.1 [**Integrated Circuits**]: Types and Design Styles – *gate arrays.*

B.7.2 [**Integrated Circuits**]: Design Aids – *placement and routing*.

General Terms
Algorithms, Measurement, Experimentation.

Keywords
pipelined FPGA, pipelined interconnect, registered routing, architecture explorations, PipeRoute

1. INTRODUCTION
Over the last few years, reconfigurable technologies have made remarkable progress. Today, state-of-the-art devices [1,14] from FPGA vendors provide a wide range of functionalities. Coupled with gate-counts in the millions, these devices can be used to implement entire systems at a time. However, improvements in FPGA clock cycle times have consistently lagged behind advances in device functionalities and capacities. Even the simplest circuits cannot be clocked at more than a few hundred megahertz.

FPGA'04, February 22–24, 2004, Monterey, California, USA.

A number of research groups have tried to improve clock cycle times by proposing pipelined FPGA architectures. Some examples of pipelined architectures are HSRA [13], RaPiD [4,6], and the architecture proposed in [11]. The distinguishing features of a pipelined FPGA are the number and location of registers in the architecture. Pipelined FPGAs provide a relatively large number of registers, both in the logic and interconnect structures. Applications that are mapped to pipelined FPGAs are often retimed to take advantage of abundant, easily available registers.

Pipelined FPGA architecture design poses a number of challenges, not the least of which is the composition of the interconnect structure. Earlier work [2,3] has shown that the design of FPGA interconnect structures involves tradeoffs amongst different parameters like segment-length, switch-box types and layout considerations. However, the interconnect structure of a pipelined FPGA is different. Unlike conventional architectures, the interconnect structure of a pipelined FPGA may include a large number of registers. The number and location of interconnect registers plays an important role in determining the performance of applications mapped to pipelined FPGAs. If the number of interconnect registers is too few, the benefits of pipelining may get lost in long, circuitous routes. On the other hand, the area penalty due to too many interconnect registers may reduce the impact of improvements in clock cycle time.

The objective of this paper is to parameterize and explore the performance of pipelined interconnect structures. Specifically, we try to answer the following questions:

- What are the benefits of registering the IO terminals of logic units? Note that both RaPiD and HSRA provide register banks at IO terminals.
- How many sites in the interconnect structure should be registered? A related question is how many registers should a single site provide?
- How long should the segments of registered routing tracks be?
- How does the flexibility of the interconnect structure affect the performance of pipelined FPGAs?

To the best of our knowledge, the only previous work that explored pipelined interconnect structures can be found in [12]. In that work, the authors present a limited study that demonstrates speed-ups by adding registers to routing switches. The authors do not explore multiple-register interconnect sites, segment lengths of registered tracks, or the flexibility of the interconnect structure. This work expands the pipelined interconnect exploration space to include these parameters.

The rest of this work is organized as follows. Section 2 introduces the concept of pipelined signals. In Section 3, we describe the target architecture that we used in our experiments. Section 4 describes the set of benchmark netlists that we selected. In Section 5, we describe the CAD tools that we used to enable the exploration. Section 6 presents the trends that we observed during our exploration. A quantitative evaluation of the results of our exploration is presented in Section 7. Finally, in Section 8 we conclude this paper and identify areas for future work.

2. PIPELINED SIGNALS

Netlists that are mapped to pipelined FPGAs generally contain a significant number of *pipelined* signals. A pipelined signal is a signal that must go through registers in the interconnect structure. An example of a pipelined signal ***sig*** is shown in Fig. 1. In this case, there must be three registers between S and K1, four registers between S and K2 and five registers between S and K3. It can easily be seen that a pipelining-unaware FPGA router may not find a route for ***sig*** that includes enough registers to individually satisfy the register constraints between the source and sinks. In [10], we showed that the even the two-terminal pipelined routing problem is NP-Complete, and proposed an algorithm that can be used to efficiently find routes that contain the requisite number of registers.

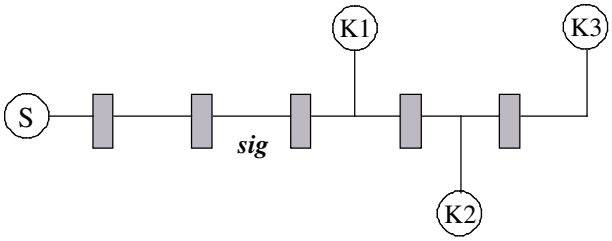

Fig. 1: A multi-terminal pipelined signal

3. THE RaPiD ARCHITECTURE

In this section we describe features of the RaPiD architecture [6]. RaPiD provides the FPGA framework used in this paper. The RaPiD architecture is targeted to high-throughput, compute-intensive applications like those found in DSP. Since such applications are generally pipelined, the RaPiD datapath and interconnect structures include an abundance of registers. The 1-Dimensional (1-D) RaPiD datapath (Fig. 2) consists of coarse-grained functional units that include ALUs, multipliers, small SRAM blocks, and general purpose registers (hereafter abbreviated GPRs). Each functional unit is 16 bits wide. The interconnect structure consists of 1-D routing tracks that are also 16 bits wide. There are two types of routing tracks: short tracks and long tracks. Short tracks are used to achieve local connectivity between functional units, whereas long tracks traverse longer distances along the datapath. In Fig. 2, the uppermost five tracks are short tracks, while the remaining tracks are long tracks. A separate routing multiplexer is used to select the track that drives each input of a functional unit. Each output of a functional unit can be configured to drive multiple tracks by means of a routing demultiplexer.

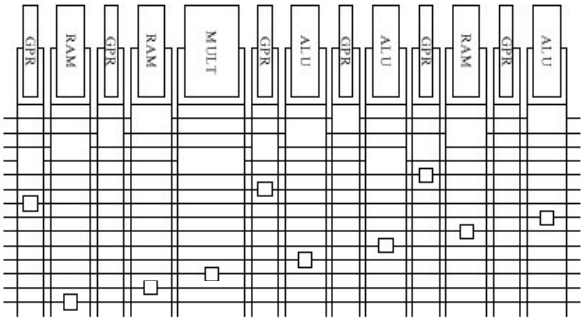

Fig. 2: An example of a RaPiD [6] architecture cell. Several RaPiD cells can be tiled together to create a representative architecture.

The long tracks in the RaPiD interconnect structure are segmented by means of bus connectors (shown as empty boxes in Fig. 2 and abbreviated BCs). BCs serve two roles in the RaPiD interconnect structure. First, a BC serves as a buffered, bidirectional switch that facilitates the connection between two long-track segments. Second, a BC serves the role of an interconnect register site. RaPiD provides the option of picking up zero, one, two or three registers at each BC. The total number of BCs determines the number of registers that can be acquired in the interconnect structure.

While BCs are used as registered, bidirectional switches that connect segments on the same long track, GPRs can be used to switch tracks. A GPR's input multiplexer and output demultiplexer allow a connection to be formed between arbitrary tracks. At the end of a placement phase, all unoccupied GPRs are included in the routing graph as unregistered switches. The ability to switch tracks provides an important degree of flexibility while routing netlists on the RaPiD architecture.

4. BENCHMARKS

The set of benchmark netlists that we used during exploration includes implementations of FIR filters, sorting algorithms, matrix multiplication, edge detection, 16-way FFT, IIR filtering and a camera imaging pipeline. While selecting the benchmark set, we tried to include a diverse set of applications that were representative of the domains to which RaPiD is targeted. We also tried to ensure that the benchmark set was not unduly biased towards netlists with too many or too few pipelined signals. Fig. 3 shows the percentage of pipelined signals in each benchmark netlist.

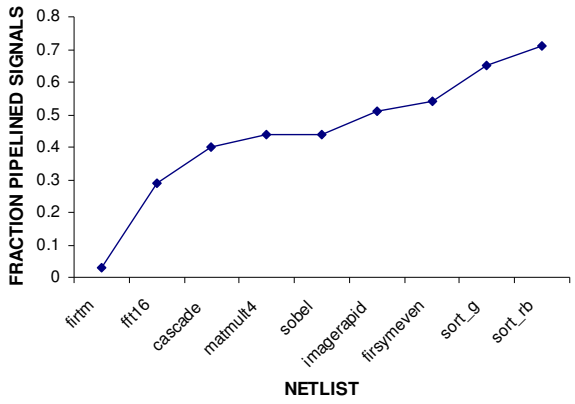

Fig. 3: The fraction of pipelined signals in each benchmark netlist.

5. CAD TOOLS & FLOW

The CAD flow that supports the exploration of RaPiD's pipelined interconnect structure is presented in this section. Applications are mapped to netlists using the RaPiD compiler [5], and the architecture is represented as an annotated structural Verilog file. During data acquisition, a flexible architecture generation tool is used to produce parameterized architectures that represent different points in the interconnect exploration space.

The placement of a netlist is determined using a Simulated Annealing [8] algorithm. The cost of a placement is formulated as a linear function of the maximum and average *cutsize*, where *cutsize* is the number of signals that need to be routed across a vertical partition of the architecture for a given placement. Since the RaPiD interconnect structure provides a fixed number of routing tracks, the cost function must be sensitive to changes in maximum cutsize. At the same time, changes in average cutsize also influence the cost of a placement. This is because average cutsize is a direct measure of the total wirelength of a placement. Pipelining information is included in the cost of a placement by mapping each pipelining register (a *pipelining* register is a register that must be mapped to an interconnect register) in the netlist to a unique BC in the interconnect structure. Our high-level objective in mapping pipelining registers to BCs is to place netlist components such that the router is able to find a sufficient number of BCs in the interconnect structure while routing pipelined signals. A more detailed discussion of the placement strategy can be found in [9,10].

After the final placement of a netlist has been determined, the netlist is routed using the PipeRoute algorithm [10]. PipeRoute is an architecture independent algorithm based on Pathfinder [7] that routes pipelined signals on to FPGAs that have a registered interconnect structure. The basic building block used by PipeRoute is an optimal 1-Register router that finds a lowest-cost two terminal route that goes through at least one register in the interconnect structure. A two terminal N-Register route is recursively built from an (N-1)-Register route by successively replacing each segment of the (N-1)-Register route by an optimal 1-Register route, and then selecting the lowest cost N-Register route. The routing tree for a multi-terminal pipelined signal is built one sink at a time. Every time a new sink is to be routed, we try to greedily reuse segments within the current, partially built routing tree to provide registers on the route to the new sink.

The PipeRoute algorithm presented in [10] was developed under certain simplifying assumptions. First, we assumed that register sites in the interconnect structure can only provide zero or one register. Second, we did not address the fact that the IO terminals of functional units may themselves be registered. As mentioned in Section 3, the BCs in RaPiD's interconnect structure allow a signal to pick up between zero and three registers. Furthermore, the outputs of each functional unit are connected to the interconnect structure through a register bank that also provides between zero and three registers. In order to take advantage of registered IO terminals and multiple-register sites in the interconnect structure, we use a pre-processing heuristic before routing a netlist. This heuristic attempts to locally maximize the number of registers that can be acquired at registered IO terminals and multiple-register sites. By doing so, we try to reduce the total number of register sites that have to be found in the interconnect structure during pipelined routing.

A last, albeit important, feature that we added to the PipeRoute algorithm was to make it timing driven. Since the primary objective of pipelined FPGAs is the reduction of clock cycle time, it is imperative that a pipelined routing algorithm maintains control over the criticality of pipelined signals during routing. In making PipeRoute timing driven, we drew inspiration from the Pathfinder algorithm. While routing a signal, Pathfinder uses the criticality of the signal in determining the relative contributions of the congestion and delay terms to the cost of routing resources. However, in the special case of a pipelined signal, the signal's route may contain multiple interconnect registers and hence different segments on the route may be at different criticalities. Furthermore, the signal's route may go through *different* interconnect registers from one routing iteration to the next. Thus, before routing a pipelined signal, we are faced with making an intelligent guess about the overall criticality of a pipelined signal. Our solution is to make a pessimistic choice and assign the criticality of the most critical segment to the criticality of the pipelined signal.

6. INTERCONNECT EXPLORATION

In this section, we present our interpretation and analysis of the trends that we observed while exploring RaPiD's pipelined interconnect structure. Our primary measure of the quality of a given point in the exploration space is the post place-and-route geometric average of the area-delay product across the benchmark set. The area-delay product of a netlist is measured from the minimum number of RaPiD cells required to route a netlist in less than thirty-two tracks. Area models for the RaPiD architecture are derived from a combination of the current layout of the RaPiD cell, and transistor-count models. The delay model is extrapolated from SPICE simulations.

Before presenting our results, we briefly explain the effects of certain important interconnect features on the area and delay of a netlist:

Track Count: The track count of a netlist is the minimum number of tracks required to route the netlist. Track count directly affects area in two ways. First, the height and width of the IO multiplexers and demultiplexers are determined by the number of tracks that connect to them. Second, the number of BCs in the architecture is directly proportional to the number of tracks.

BCs: The frequency and number of BCs in the interconnect structure affects both area and delay. A large number of BCs provide an abundance of interconnect register sites. Consequently, a BC-rich interconnect structure improves the routability of pipelined signals. Routability improvements generally result in track count reductions, and if the area benefit due to such reductions is greater than the area-penalty of a large number of BCs, an overall area win may result. The number and location of BCs in the interconnect structure also influences the delay characteristics of a netlist. One reason is the effects of segmentation on the critical path delay of a netlist [2]. Another reason is that the number of BCs determines the quality of the routes of pipelined signals. Recall that the number of BCs is a direct measure of the number of interconnect registers. In BC-poor architectures, the pipelined router finds long, circuitous routes for heavily pipelined signals. Such poor-quality routes result in a deterioration of the delay characteristics of a netlist.

The remainder of this section is devoted to an axis-by-axis exploration of RaPiD's pipelined interconnect structure.

6.1 REGISTERED IO TERMINALS:

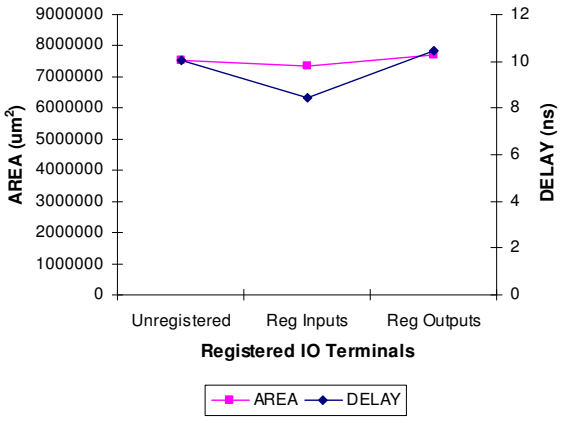

Fig. 4: Area and delay numbers for architectures with registered outputs, registered inputs and unregistered IO terminals.

Our first step is to explore the possible benefits of functional units that have 'registered' input or output terminals. An IO terminal of a functional unit is registered if the terminal can be connected to the interconnect structure through a local register bank. Local register banks allow pipelined signals to pick up registers at the functional unit, thus reducing the number of registers that have to be found in the interconnect structure. Fig. 4 shows the area and delay numbers that we obtained on mapping the benchmark netlists to architectures with registered input, registered output and unregistered terminals. Surprisingly, the effect of registered IO terminals on area is negligible. This is because the area penalty of adding registers to IO terminals nullifies the area benefits attributable to the track count reductions shown in Fig. 5.

While area is insensitive to registered IO terminals, the delay performance of architectures with registered inputs is clearly better. This is because the preprocessing heuristic mentioned in Section 5 moves a large number of registers from the interconnect structure in to the inputs of functional units. Consequently, the pipelined router has to find fewer registers in the interconnect structure, thus improving the delay characteristics of netlists. Interestingly, architectures with registered outputs show no delay improvement when compared to architectures that have unregistered IO. This is probably because the number of interconnect registers that are moved in to the outputs of functional units is an insignificant fraction of the total number of interconnect registers that have to be found during pipelined routing. Overall, architectures with registered input terminals proved to be the best choice in terms of area-delay product.

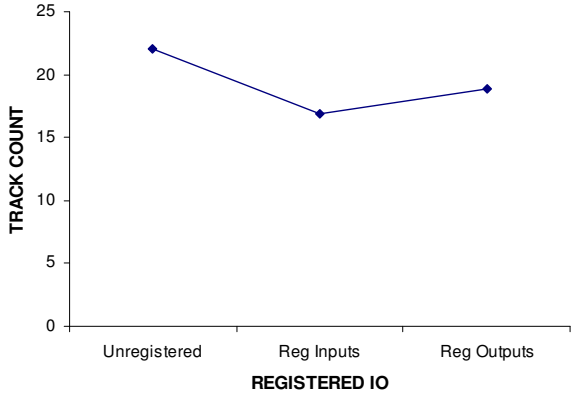

Fig. 5: Track counts for architectures that have registered input, registered output and unregistered IO terminals.

6.2 BUS CONNECTORS:

In Section 3 we mentioned that BCs serve as buffered registered switches in the RaPiD interconnect structure The total number of BCs in the interconnect structure plays a major role in determining the overall area and delay of a netlist mapped to the RaPiD architecture. The number of BCs in the interconnect structure is varied by changing the number of BCs per long track in a RaPiD cell. (Hereafter, 'BCs per long track' will simply be called BCs per track). For example, the RaPiD cell shown in Fig. 2 has one BC per track, while the cell shown in Fig. 6 has two BCs per track. Varying the number of BCs per track not only changes the number of interconnect register sites, but also the length of long track segments. Long track segments in Fig. 2 span thirteen functional units, while long track segments in Fig. 6 span six or seven functional units.

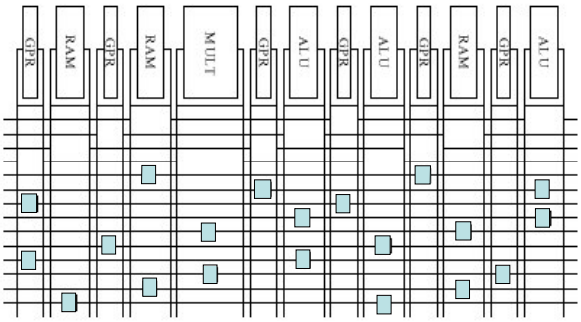

Fig. 6: A RaPiD cell that has two BCs per long track.

Fig. 7 shows the area and delay numbers that we obtained as a result of varying the number of BCs per track (the number 0.5 on the x-axis implies architectures that had a single BC per track for every two RaPiD cells). There is a marked improvement in delay when going from half to a single BC per track. This is because at half BC per track, track segments are too long and there are relatively few BCs available for pipelined signals. On increasing the number of BCs per track past one, the delay gradually goes back up. This is because the delay incurred in traversing an increased number of BCs along a long track more than offsets improvements due to shorter track segments and tighter pipelined routes.

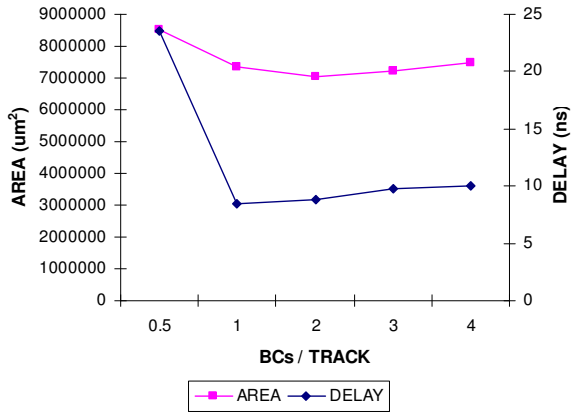

Fig. 7: The effect of varying number of BCs per track on area and delay.

Fig. 7 also shows an area benefit as the number of BCs per track is increased to two. This is consistent with the 45% reduction in track count when the number of BCs per track is increased from half to two (Fig. 8). The area gradually increases after that due to the fact the area-cost of adding more BCs per track exceeds any improvements in track count.

Fig. 9 shows the area-delay product trend that we obtained. The area-delay products at one and two BCs per track are within 1% of each other, which leads us to believe that anywhere between one and two BCs per track is a good architectural choice.

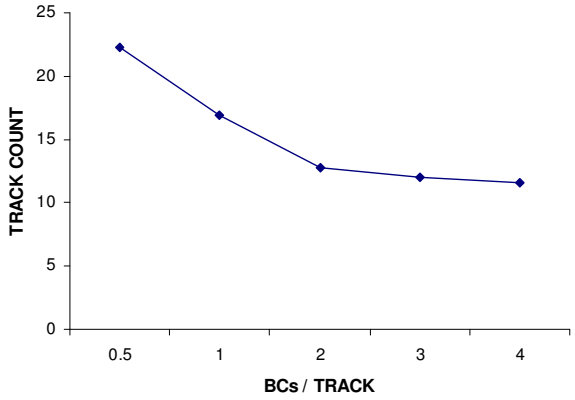

Fig. 8: The effect of varying number of BCs per track on track count.

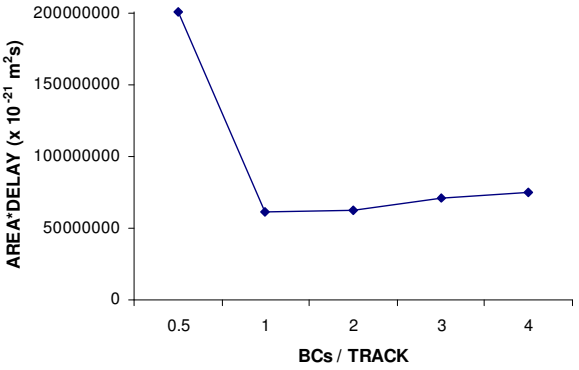

Fig. 9: The effect of varying number of BCs per track on the area-delay product.

6.3 MULTIPLE-REGISTER BUS CONNECTORS:

The number of registers in a BC is another parameter that influences the overall area-delay performance of a circuit. An increase in the number of registers per BC allows pipelined signals to pick up a greater number of registers at a single interconnect site. This improves track count because a reduced number of BCs have to be found while routing pipelined signals. At the same time, the delay characteristics of the netlists may also get better due to a reduction in the long, circuitous routes that are found while routing pipelined signals on architectures that have register-poor BCs. Fig. 10 shows area and delay trends when the number of registers per BC is varied between one and seven.

There is an improvement in area as the number of registers per BC is increased to three. However, the area goes back up as the number of registers per BC is increased past that point. This is because increases in BCs area exceed any area improvements attributable to track-count reductions (Fig. 11).

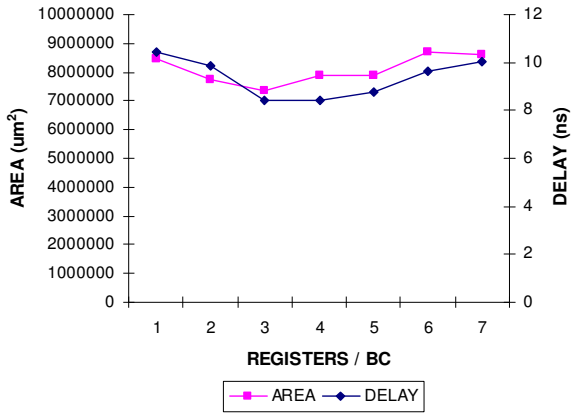

Fig. 10: The effect of varying number of registers per BC on area and delay.

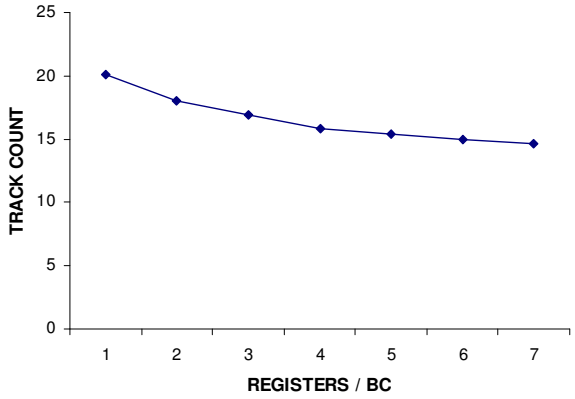

Fig. 11: The effect of varying number of registers per BC on track count.

At first sight, the delay trend in Fig. 10 seems surprising. While there is an expected improvement in delay as the number of registers per BC is increased to four, the delay unexpectedly goes back up past that point. A possible reason for this behavior is the greedy manner in which the preprocessing heuristic pushes interconnect registers into functional unit input terminals. While conducting experiments, we assume that the number of registers in a BC is equal to the maximum number of registers that can be picked up at the inputs of functional units (we made this assumption to limit the number of axes that we explored to a practical number). Thus, if the number of registers per BC is large, so is the number of registers that can be moved into the sinks of a pipelined signal. A shortcoming of this assumption is that long segments of a pipelined signal may get unpipelined because of the removal of registers from the interconnect structure. This phenomenon is illustrated in Fig. 12. Assume that a maximum of four registers can be picked up at the sinks K1-K8. In this case, one interconnect register will be moved into K1, two into K2, three into K3, and four into K4-K8. This process effectively unpipelines a long-track segment, which in turn may increase the critical path delay of a netlist.

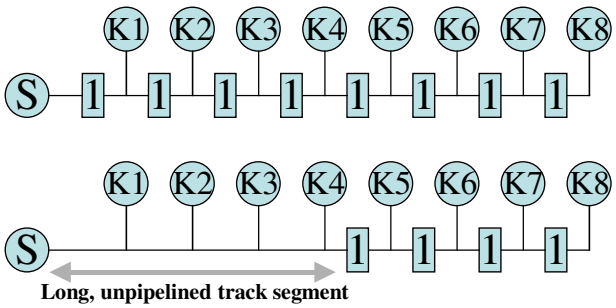

Fig. 12: Pushing registers from the interconnect structure into functional unit inputs sometimes results in long, unpipelined track segments.

Fig. 13 shows the area-delay product vs. number of registers per BC. A clear sweet spot can be observed at three registers per BC.

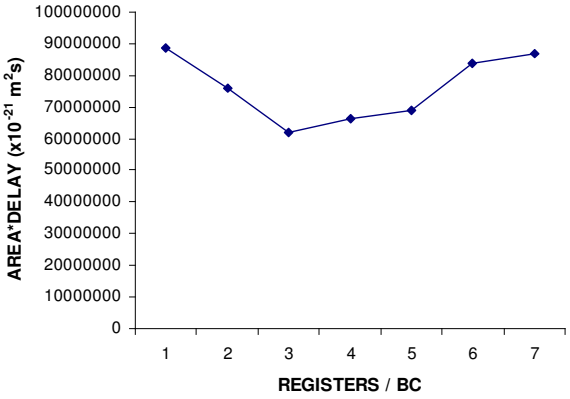

Fig. 13: The effect of varying number of registers per BC on the area-delay product.

6.4 SHORT / LONG TRACK RATIO:

RaPiD's interconnect structure is a mix of short tracks and long tracks (Fig. 2). Short tracks achieve local connectivity between functional units. Long tracks are used to traverse longer distances along the datapath, and are segmented by means of BCs. In addition to serving as bidirectional switches, BCs also play the role of interconnect register sites.

We demonstrated earlier that the combined area-delay product of the benchmark netlists is sensitive to the number of BCs per track. Varying the number of BCs per track changes the distribution and total number of BCs in the interconnect structure. Another factor that directly affects the number of BCs is the ratio between short and long tracks. Fig. 14 shows the area and delay trends that we observed on varying the fraction of short tracks in the architecture. Notice that the delay is higher for architectures that have short-track fractions < 0.28. This trend may be due to the fact that short-track poor architectures force signals to use long-track segments to establish connections that could otherwise have been routed on short-track segments[1].

[1] In general, the routing delay of a long-track segment exceeds that of a short-track segment. A long segment has more resistance due to its length, and greater fanout capacitance.

For short-track fractions > 0.28, the delay again increases because of two reasons. First, long-track poor architectures force signals to use multiple short-track segments to establish connections that may have otherwise used a single long-track segment[2]. Second, the reduction in the number of BCs increases the possibility of long, circuitous routes being found for heavily pipelined signals.

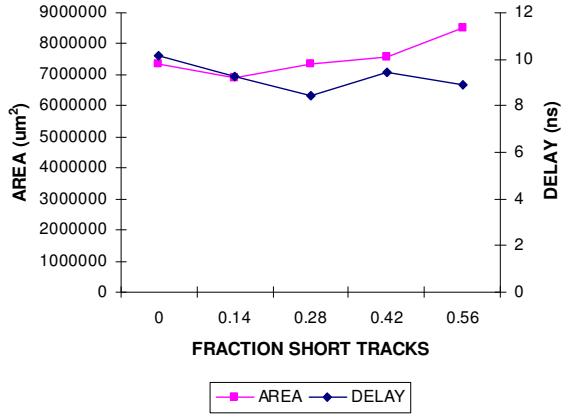

Fig. 14: The effect of varying fraction of short tracks on area and delay.

The area curve has a minimum at 0.14. Architectures that are relatively poor in short tracks pay an area penalty due to an excessive number of BCs and an increased track count (Fig. 15). The track count increases because signals that could have been routed on segments on the same short track have to use segments on *different* long tracks. As the short-track fraction is increased past 0.14, the area goes back up. This is again due to an increase in track count (Fig. 15). This time however, the track count increases because fewer BCs are available to pick up registers in the interconnect structure.

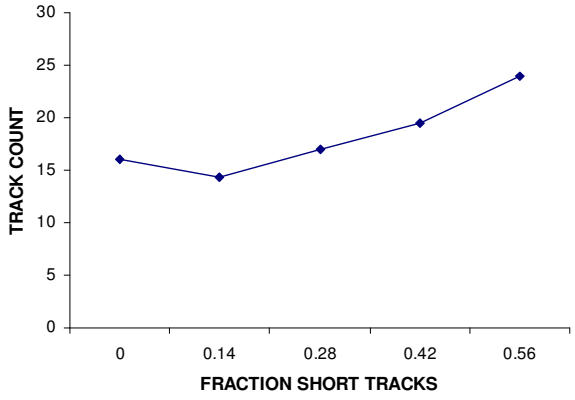

Fig. 15: The effect of varying fraction of short tracks on track count.

The area-delay trend vs. the fraction of short tracks in Fig. 16 shows a clear minimum at 0.28.

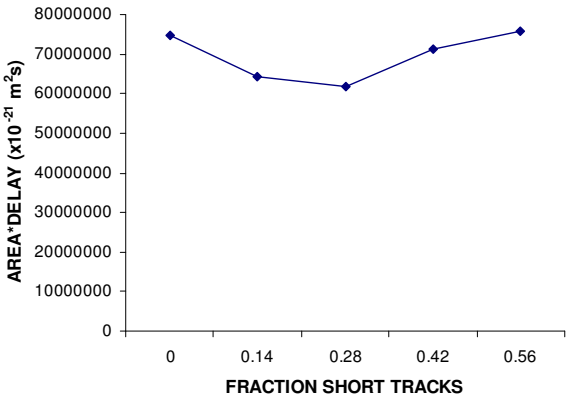

Fig. 16: The effect of varying fraction of short tracks on the area-delay product.

6.5 DATAPATH REGISTERS (GPRs):

The main purpose of GPRs in the RaPiD architecture is to serve as pipelining sites in the datapath structure. However, any unoccupied GPR units can also be used by signals to switch tracks in the interconnect structure. A large number of unoccupied GPRs in the datapath structure increases the flexibility of the interconnect structure. Consequently, the total number of GPRs in the architecture plays a role in determining the routability of netlists that are mapped to the RaPiD architecture. This role may be especially pronounced in netlists that occupy a large percentage of GPRs in the datapath. Fig. 17 shows area and delay trends when the number of GPRs per RaPiD cell is varied between five and ten (the number 6 on the x-axis corresponds to the number of GPRs provided in the original RaPiD cell shown in Fig. 2).

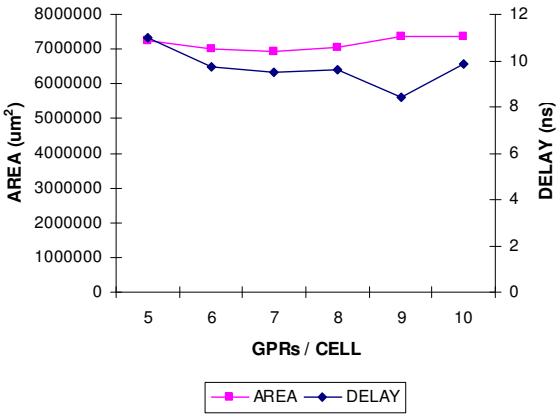

Fig. 17: The effect of increasing the number of extra GPRs / RaPiD cell on area and delay.

[2]Note that unoccupied GPRs in the datapath can be used by signals to switch tracks arbitrarily.

Fig. 17 shows that varying the number of GPRs per RaPiD cell produces marginal area benefits while going from five to seven GPRs per cell. This is consistent with the reduction in track count shown in Fig. 18. When the number of GPRs / cell is increased past seven, the area goes back up due to the penalty of adding extra GPRs to the architecture. Notice in Fig. 18 that track count remains relatively constant past seven GPRs.

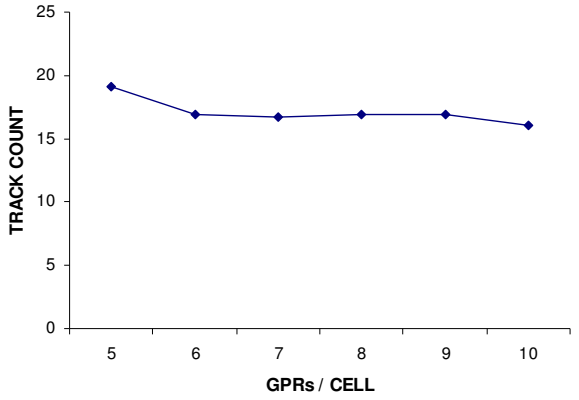

Fig. 18: The effect of increasing the number of extra GPRs / RaPiD cell on track count.

The delay curve in Fig. 17 has a minimum at nine GPRs per cell. Architectures that have fewer than nine GPRs per cell do not have sufficient switching sites. Consequently, the pipelined router is forced to find potentially longer routes for pipelined signals. The delay goes back up past nine GPRs per cell because the delay of track segments increases. This increase can be attributed to the greater fanout capacitance per segment that results when the number of GPRs per cell is increased. Fig. 19 shows that the area-delay product is minimum for architectures that have nine GPRs per RaPiD cell.

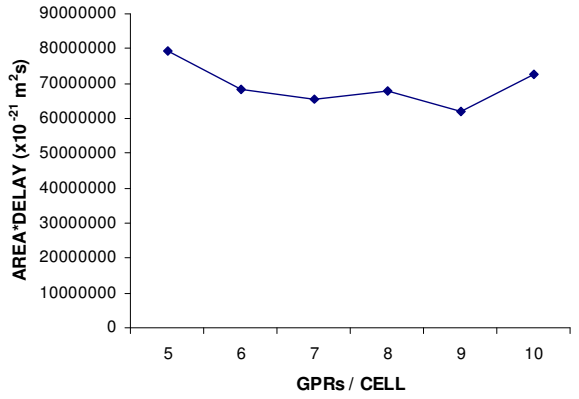

Fig. 19: The effect of increasing the number of extra GPRs / RaPiD cell on area-delay product.

7. QUANTITATIVE EVALUATION

In this section we present a quantitative evaluation of our results. First, we quantitatively compare the original RaPiD architecture with the results of the exploration of the interconnect structure. Second, we quantify the area overhead incurred in placing and routing benchmark netlists on the post-exploration architecture that we found.

7.1 COMPARISONS WITH RaPiD

We reproduce the RaPiD cell from Section 3 in Fig. 20. The RaPiD cell has registered outputs, a single BC per track, three registers per BC and 28% short tracks.

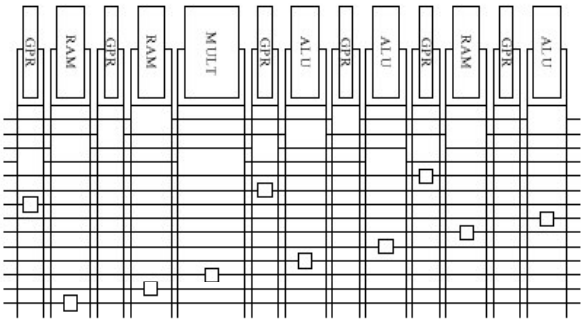

Fig. 20: An illustration of the RaPiD cell [6].

We first note that the choices of a single BC per track, three registers per BC and 28% short tracks are in fact consistent with the findings of our exploration in Section 6. At the same time, there are differences between RaPiD and our findings. First, RaPiD has registered outputs. Our exploration found that registered inputs are a better choice. Second, we found that the number of GPRs per RaPiD cell is insufficient, and that there should be nine GPRs per RaPiD cell (three more than the six GPRs shown in Fig. 20). Table 1 presents a comparison between the original RaPiD architecture and the best post-exploration architecture that we found. Column 1 lists the benchmark netlists, column 2 lists area-delay products (all area-delay product values are $x10^{-21}m^2s$) measured from the post-exploration architecture, column 3 lists area-delay products measured from RaPiD, column 4 lists percentage improvements, and column 5 lists the fraction of pipelined signals in each netlist. RaPiD outperforms the post-exploration architecture for netlists that have less than 30% pipelined signals, while the post-exploration architecture performs better than RaPiD for netlists that have more than 54% pipelined signals. Overall, the post-exploration architecture's area-delay product is 19% better.

Table 1: A quantitative comparison of RaPiD with the post-exploration architecture

Netlist	Post Explore	RaPiD	% Improve	Fraction Pipelined
sort_g	53315010	42731038	- 25%	0.03
sort_rb	69861704	58841648	- 19%	0.29
matmult4	124208334	131567582	+ 6%	0.4
firtm	43718308	69887073	+ 37%	0.44
firsymeven	149055072	111187648	- 34%	0.44
fft16	108339199	90049942	- 20%	0.51
imagerapid	66659262	102737262	+ 35%	0.54
cascade	19128011	41412889	+ 54%	0.65
sobel	31809351	88791876	+ 65%	0.71
GEOMEAN	**61850995**	**76281065.3**	**+ 19%**	

7.2 AREA OVERHEAD

We quantify area overhead by comparing the area of a netlist mapped to the post-exploration architecture with the area of a netlist that is mapped to the same architecture using a pipelining-unaware place-and-route flow. In a pipelining-unaware flow, pipelined signals are treated like normal, unpipelined signals. Fig. 21 shows the unpipelined version of a pipelined signal.

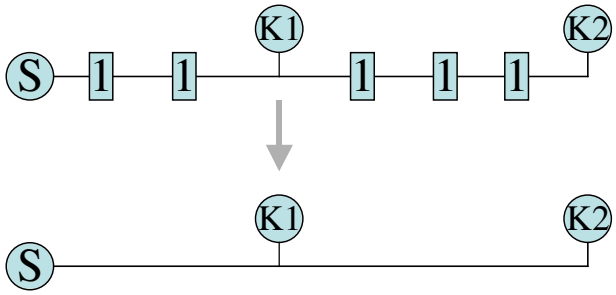

Fig. 21: Unpipelining a pipelined signal.

The pipelining-unaware placement tool attempts to reduce only maximum and average cutsize (Section 5). The pipelining-unaware router attempts only connectivity routing, since there are no registers to be found in the interconnect structure. Fig. 22 shows the area overhead incurred in placing and routing benchmark netlists on the post-exploration architecture. The overall area overhead incurred is 18%.

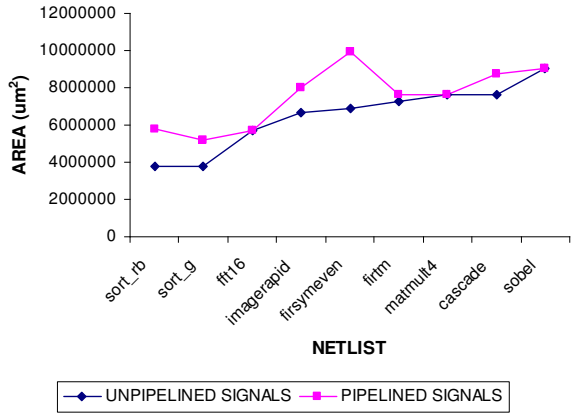

Fig. 22: Area overhead incurred in placing and routing the benchmark netlists on the post-exploration architecture.

8. CONCLUSIONS & FUTURE WORK

The primary objective of this work was to identify and explore various *interconnect* parameters that affect the overall performance of applications that are mapped to pipelined FPGA architectures. Our hope is that the designers of pipelined FPGAs will use the findings of our exploration as an aid in the future. In conclusion:

1. Adding registers to the inputs of functional units may improve the performance of pipelined netlists (Section 6.1). However, if the number of registers is large, greedily pushing the maximum number of registers into inputs may result in a deterioration of the delay of a netlist (Section 6.3).
2. The number and distribution of registered interconnect sites greatly influence overall performance. If there is an insufficient number of interconnect register sites, the pipelined router is forced to find long, circuitous routes that adversely affect both track count and delay (Section 6.2). On the other hand, peppering the interconnect structure with register sites may result in an unacceptable area penalty.
3. For reasons similar to those in 2, the number of registers per interconnect site also has to be carefully selected (Section 6.3).
4. The flexibility of the interconnect structure has a bearing on the performance of netlists. In Section 6.5, we show that architectures that are GPR-poor do not perform well. This is because of increased track counts and longer pipelined routes. On the other hand, architectures that have too many GPRs suffer from an excessive area-penalty.

There are a number of areas that we may investigate in future work. One area that we may research is algorithms for timing-driven pipelined routing. A second area that we could look at is the development of heuristics that can efficiently utilize multiple-register sites in the interconnect structure of pipelined FPGAs. Finally, it would be useful to explore the area and delay performance of island-style, pipelined FPGA architectures.

9. ACKNOWLEDGMENTS

We would like to thank Chris Fisher for providing us with a set of benchmark netlists. Thanks are also due to Ken Eguro and Shawn Phillips for the area and delay models. This work was supported by grants from the NSF. Scott Hauck was supported in part by an NSF Career Award and an Alfred P. Sloan Fellowship.

10. REFERENCES

[1] Altera Inc., "Stratix™ Device Family Features", available at http://www.altera.com.

[2] V. Betz and J. Rose, "FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density", *ACM/SIGDA Seventh International Symposium on Field-Programmable Gate Arrays*, pp 59-68, 1999.

[3] V. Betz and J. Rose, "Circuit Design, Transistor Sizing and Wire Layout of FPGA Interconnect", *IEEE Custom Integrated Circuits Conference*, pp 171 - 174, 1999.

[4] Darren C. Cronquist, Paul Franklin, Chris Fisher, Miguel Figueroa, and Carl Ebeling, "Architecture Design of Reconfigurable Pipelined Datapaths", *Twentieth Anniversary Conference on Advanced Research in VLSI,* pp 23-40, 1999.

[5] Darren C. Cronquist, Paul Franklin, Stefan Berg, Carl Ebeling, "Specifying and Compiling Applications to RaPiD", *Field-Programmable Custom Computing Machines,* 1999.

[6] Carl Ebeling, Darren C. Cronquist, Paul Franklin, "RaPiD - Reconfigurable Pipelined Datapath", *6th International Workshop on Field-Programmable Logic and Applications,* pp 126-135, 1996.

[7] Larry McMurchie and Carl Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs", *ACM Third International Symposium on Field-Programmable Gate Arrays*, pp 111-117, 1995.

[8] C. Sechen, *VLSI Placement and Global Routing Using Simulated Annealing*, Kluwer Academic Publishers, Boston, MA: 1988.

[9] A. Sharma, "Development of a Place and Route Tool for the RaPiD Architecture", *Master's Project, University of Washington*, December 2001.

[10] A. Sharma, C. Ebeling, S. Hauck, "PipeRoute: A Pipelining-Aware Router for FPGAs", *11th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp 68-77, 2003.

[11] Amit Singh, Arindam Mukherjee, Malgorzata Marek-Sadowska, "Interconnect Pipelining in a Throughput-Intensive FPGA Architecture", *ACM/SIGDA Ninth International Symposium on Field-Programmable Gate Arrays*, pp 153-160, 2001.

[12] Deshanand P. Singh, Stephen D. Brown, "The Case for Registered Routing Switches in Field Programmable Gate Arrays", *ACM/SIGDA Ninth International Symposium on Field-Programmable Gate Arrays*, pp 161-169, 2001.

[13] William Tsu, Kip Macy, Atul Joshi, Randy Huang, Norman Walker, Tony Tung, Omid Rowhani, Varghese George, John Wawrzynek and Andre DeHon, "HSRA: High-Speed, Hierarchical Synchronous Reconfigurable Array", *ACM Seventh International Symposium on Field-Programmable Gate Arrays*,pp , 1999.

[14] Xilinx Inc., "VirtexII™ Platform FPGA Features", available at http://www.xilinx.com.