Innovate or Perish: FPGA Physical Design

Taraneh Taghavi, Soheil Ghiasi
University of California,
Los Angeles
{taghavi, soheil}@cs.ucla.edu

Abhishek Ranjan, Salil Raje
Hier Design Inc.
{ranjan, salil}@hierdesign.com

Majid Sarrafzadeh
University of California,
Los Angeles
majid@cs.ucla.edu

ABSTRACT
The recent past has seen a tremendous increase in the size of design circuits that can be implemented in a single FPGA. The size and complexity of modern FPGAs has far outpaced the innovations in FPGA physical design. The problems faced by FPGA designers are similar in nature to those that preoccupy ASIC designers, namely, interconnect delays and design management. However, this paper will show that a simple re-targeting of ASIC physical design methodologies and algorithms to the FPGA domain will not suffice. We will show that several well researched problems in the ASIC world need new problem formulations and algorithms research to be useful for today's FPGAs. Partitioning, floorplanning, placement, delay estimation schemes are only some of the topics that need complete overhaul. We will give problem formulations, motivated by experimental results, for some of these topics as applicable in the FPGA domain.

Categories and Subject Descriptors
B.7.1 [Types and Design Styles]: Gate Arrays; B.7.2 [Design Aids]: Layout, Placement and Routing.

General Terms
Algorithms, Design, Experimentation.

Keywords:
Physical Design, FPGA, Routing Architecture, Partitioning, Floorplanning, Placement, Delay Estimation.

1. INTRODUCTION
Advances in process technology today are enabling a profound increase in the number of applications that can be realized using FPGAs. Multi-million gate capacity (Figure 1.1) and clock speeds approaching 400 MHz (Figure 1.2) for FPGA-based design are now main-stream. Densities approaching 10 million gates, shorter design cycles and reduced development costs make programmable devices increasingly attractive for a broader range of applications; from networking, telecommunications and medical devices to other consumer products.

ISPD'04, April 18-21, 2004, Phoenix, Arizona, USA.

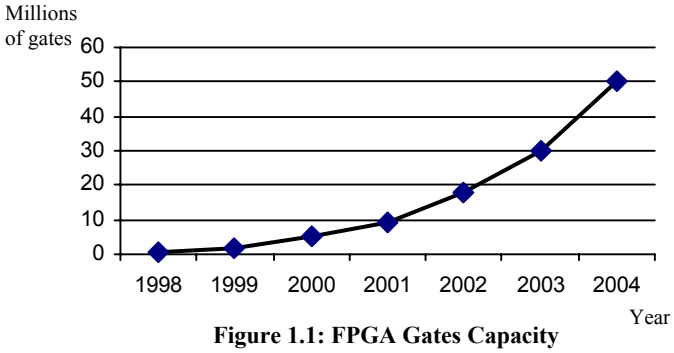


Figure 1.1: FPGA Gates Capacity

Difficult design problems associated with interconnect delay on large designs are now being seen. As witnessed when high gate-count deep sub-micron ASIC designs first emerged, interconnect could account for as much as 70-90% of overall circuit delay as critical dimensions shrink below 0.18um. These large design sizes also significantly impact cycle time due to software runtimes and an increased number of performance based iterations.

The most prudent approach to solving the FPGA design problems would be to look for guidance in the ASIC domain. The 3 developments in the ASIC design flows that have helped the ASIC designers solve the design complexity and the interconnect delay problems can broadly be categorized into:

1. Hierarchical Design Flows
2. Early Estimation and Analysis tools
3. Physical Synthesis

Hierarchical design flows entail partitioning the design into smaller and more manageable pieces, apportioning temporal (timing budgeting) and physical constraints (floorplanning), implementing each piece separately and then assembling all pieces together. Till date there has not been much attention paid to the impact of such flows on the quality of results. In [27] authors have quantified the loss in quality in using various partitioning and placement schemes during hierarchical design.

Modern placement techniques depend on partitioning algorithms to minimize wire length, for e.g., DRAGON [24], CAPO [2]. All partitioning techniques have a single area balancing constrain. FPGA architectures, however, incorporate heterogeneous resources. This places additional constraints on the balancing criteria and new algorithms need to be developed to address these new constraints.

Floorplanning, a heavily researched topic in the ASIC world, will need immense re-engineering to account for the latest FPGA architectures. For one thing, the FPGA floorplanning problem is a fixed outline problem. There are discrete and distributed resources spread across the FPGA fabric. These make all current floorplan representations and algorithms ineffective at best.

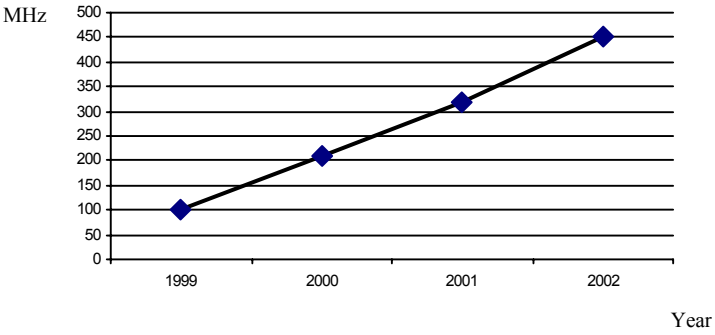


Figure 1.2: FPGA Maximum Clock Speeds

Most known interconnect estimation algorithms would fail to capture the peculiarities of the placement and routing architectures in these new FPGA fabrics. Interconnect estimation is key to physical synthesis approaches that try to predict and fix interconnect delay problem. Interconnect delay has larger impact in FPGAs than in ASICs given the large delays on the programmable switches along the route structures.

This paper will take a fresh look at the newer FPGA architectures.

We will look at the ASIC EDA algorithms and make observations as to the validity of these in the FPGA domain. The paper should also spur researchers into designing new algorithms specifically for FPGA.

2. FPGA ARCHITECTURE

Until recently FPGAs were laid out as rows and columns of **C**onfigurable **L**ogic **B**locks (CLBs). With increasingly complex requirements put on the FPGA industry the latest architectures have become increasingly heterogenous.

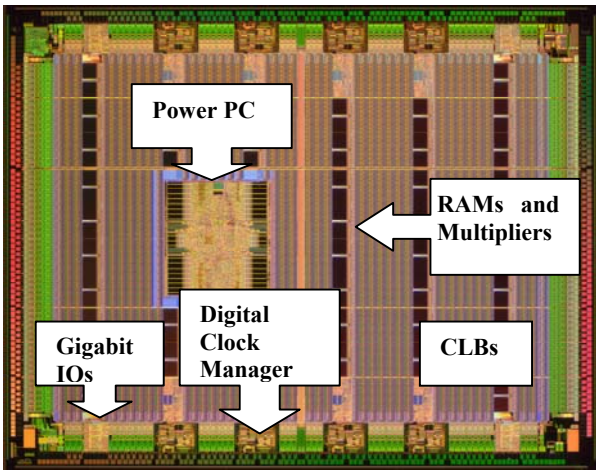


Figure 2.1: Xilinx VirtexII-Pro Device

Figure 2.1 shows one of the latest generation FPGA devices. It's a Xilinx VirtexII-Pro device. VirtexII-Pro devices are platform FPGAs designed as custom integrated circuits in a 0.13 micron process technology. They use a combination of nine layers of all copper interconnection and low k dielectric and are being produced on the latest 300 mm wafers. Flip-chip packaging is used to achieve package pin counts of up to twelve hundred pins. They have up to 4 Power PC 405 cores in them running at 300MhZ. They have up to 24 Gigabit IOs. Dedicated 10Kbit Block Rams that total up to 10MBits on chip memory. These include powerful CLBs that contain up to 8 4-input Look-up tables. They have many dedicated 18-Bits X 18-Bits pipelined multipliers running at 200MhZ for DSP applications. This device also includes Digital Clock Managers for skew reduction and stable clock generation. The trend seems to be towards adding more and more dedicated cores within a programmable fabric.

Also, noteworthy are the type of routing topologies that have been introduced by recent FPGA architectures.

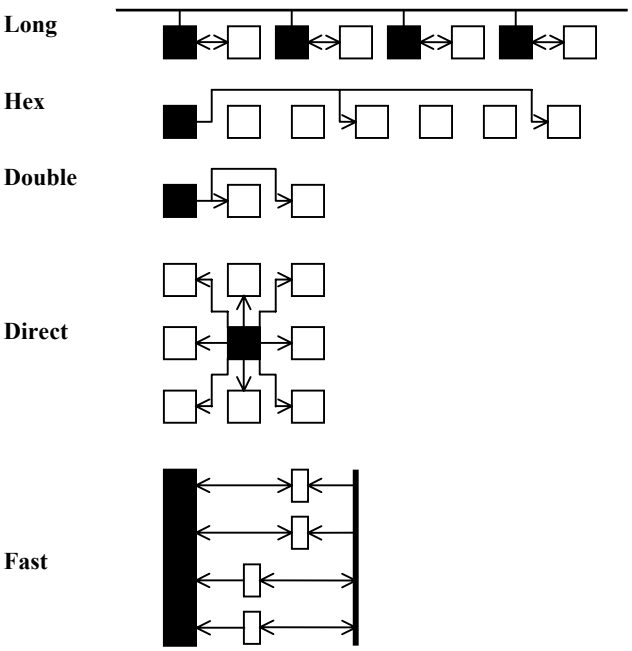


Figure 2.2: Xilinx VirtexII Device Route

Figure 2.2 (white squares denote CLBs and black squares denote route switch boxes) shows types of routes that occur in Virtex-II family of Xilinx devices. *Long lines* span full height and width of the chip, *Hex lines* route signals to every third or sixth CLB in all four directions, *Double lines* route signals to every first or second CLB in all four directions, *Direct lines* connect signals to neighboring blocks and the *Fast lines* are internal CLB local connections. The *Hex, Double,* and *Direct* have very similar delays, implying that the net delays are not directly proportional to the distance. For example, a net that spans 6 CLBs may have the same delay as the one that spans only 2 CLBs.

Most EDA algorithms that target FPGA physical design tools assume an island style FPGA with delay on routes proportional to the distance.

The following sections will detail the implications of such complex FPGA placement and routing architectures to the physical design space.

3. SIMPLIFIED ROUTING MODEL

As discussed earlier, FPGA layout and routing architecture is completely different from that of ASIC designs. One of the key differences between the two of FPGA and ASIC layouts is the existence of so called *long* nets in FPGAs that connect distant CLBs with almost no delay penalty. On the contrary, correlation of the physical distance and delay between two nodes is an underlying fact in ASIC designs.

Hence, FPGA physical design tools need to be aware of the target layout. In this section, we propose a simplified segmented routing model that can serve as the underlying layout for many FPGA physical design tools. We strive to keep the model as simple as possible, yet general and powerful enough to capture the fundamental routing characteristics of many current and future FPGA families.

A simple segmented routing model can be generally specified by a set of pairs (t_i, c_i), where t_i represents the wire type and c_i denotes the wire count (the number of wires of type t_i in the layout). Of course such a model does not completely capture all the details of FPGA architectures. For example, a hex line of Xilinx VirtexII devices can be used to connect CLBs of distance three *or* six. This fact is not modeled by our simple representation. We believe, however, that the type and count representation is simple yet powerful enough to facilitate and accommodate the development of efficient FPGA physical design tools.

Accuracy and simplicity are two contradictory dimensions of any model solution space. A reasonable tradeoff between these two dimensions provides a model that is simple enough to be analyzed, yet accurate enough to allow reasonable optimizations. The general *type and count model* can be finely tuned to correlate as much as possible with any particular FPGA architecture. On the other hand, the model can also be parameterized to simplify it even further, without harming its accuracy significantly.

Specifically, the size of nets existing in many FPGA routing architectures, seem to follow a semi-linear pattern of variation. This would allow us to specify the types of the nets in our model using two parameters. Similarly, the number of wires of each type tends to vary by a multiplicative factor compared to the wire of the immediately smaller size. Therefore, wire counts can also be approximated by two parameters, where one parameter would represent the count of wires of type 1, and the other parameter is the multiplicative factor.

More specifically, let $MR^j(a, d, q)$ denote the following set of type and count pairs:

$$MR^j(a,d,q) = \{(i, d.q^{i-1}) \mid i \in Z \;\&\; \forall 1 \le i \le j\} \cup$$
$$\{(a.(i+1-a), d.q^{i-1}) \mid i \in Z \;\&\; \forall a \le i \le j\}$$

Figure 3.1 shows the idea of our parameterized model. Parameters j and a are utilized to approximate the type of nets existing in a particular layout with two lines. Particularly, parameter j corresponds to different generations of FPGA devices. Advances in technology allow integration of a larger variety of net types on the chip, which can be modeled with incrementing j. Note that an architecture modeled by a particular value of j contains all the nets that can be represented by smaller values of j. parameters d and q contribute to the number of wires of each type.

Note that the presented model is completely based on observing current FPGA architectures, and the trend that seem to be continued for future generations of programmable devices. We believe that the presented simplifications lead to a reasonable extent of compromise between accuracy and simplicity of the model. For example $MR^j(3, 3, 3)$ can approximate the routing architecture of Xilinx Virtex II devices reasonably.

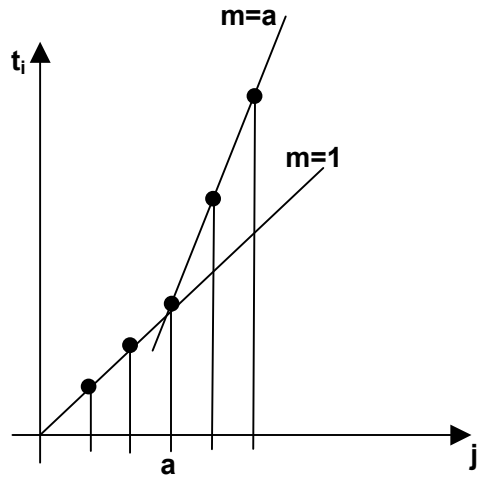


Figure 3.1. Parameter *j* corresponds to different generations of the FPGA device. Two lines with slopes of 1 and *a* are used to approximate the growth in the length of nets. There are $d.q^{(j-1)}$ nets of type *j*.

Note that j=1 always creates the routing architecture of an ordinary mesh, where each node is connected only to its immediate neighbors. Efficient FPGA physical design tools, however, need to be able to handle larger values of j, since they more closely capture the characteristics of the FPGA layouts.

4. PARTITIONING

As can be seen from our previous section modern FPGA architectures incorporate heterogeneous resources. These heterogeneous resources are distributed through the fabric of an FPGA. Many modern placement techniques are based on partitioning as their backbone, e.g. DRAGON [24], etc. At each step of these placement algorithms the placement region is divided into smaller sub-regions. The circuit is then partitioned into smaller sub-circuits and assigned to the sub-regions. The partitioning step balances the areas of the sub-circuits to conform to the areas of the regions they are assigned to. The sub-region with a smaller sub-circuit assigned to it is now a smaller instance of the original placement problem to be acted upon in the next placement step. The process continues until the regions are small enough and a detailed placement phase follows. For these placement techniques to be applicable in the FPGA space, partitioning techniques need to account for a balancing of resources with distinct types. Partitioning of circuits into sub-circuits need to balance resources used in each sub-circuit to the available resources in each of the sub-regions. For example, a partitioning solution that divides the circuit up into 2 sub-circuits such that all Look-Up-Tables are assigned to one sub-circuit and all RAMs are assigned to the other sub-circuit may have over-

subscribed these resources even if the 2 sub-circuits are balanced in terms of the total number of cells. This problem was studied by [26]. Researchers in [26] detailed an extension of hMetis to account for such resource balancing. We describe below the problem addressed by [26] and encourage researchers to attempt other algorithms that would better the quality of results.

Problem 4.1: Let the resource type of a particular cell be specified by t(v). Let cl_i^j denote the minimum number of resources of type i allowed in partition j. Then the multi-resource bisection P of hypergraph G seeks to minimize the cut subject to:

$$cl_i^1 \leq \sum_{\forall v \in V:\, P[v]=1 \text{ and } t(v)=I} 1 \leq cu_i^1 \text{ and}$$

$$cl_i^2 \leq \sum_{\forall v \in V:\, P[v]=2 \text{ and } t(v)=I} 1 \leq cu_i^2$$

5. FLOORPLANNING

Traditionally the floorplanning problem has been defined to be an area packing problem on a set of modules; some that are hard macros and have fixed height and width, others that are soft modules that have a range of aspect ratios and areas to choose from. Techniques involve having a compact representation of a floorplan [9,10,15,16,17,18,19,20,21] and using simulated annealing moves to modify the floorplan. There has been some work done for fixed outline floorplanning [22] but most of these are extensions of the area packing formulations.

Obviously, the traditional floorplan representations and techniques will not be sufficient to meet the needs of FPGA. For one, the traditional floorplan representations assume a continuous space and all locations are available for placement of modules. Secondly, the techniques assume a single area resource requirement that is a continuous function of height and width making the module areas additive.

The FPGA floorplan could be represented using the location of the lower left corner of the module and the shape of the module.

If $S = \{(I, J) | 1 \leq I \leq$ *Number of columns in device,* $1 \leq J \leq$ *Number of rows in device*$\}$, and M is a set of modules to be floorplanned, then the set of all possible module placements can be represented as $P = \{(m, l, s) | m \in M, l \in S, s \in S\}$, where m is the module, l is the location of the lower left corner of the module and s is the bounding box height and width of the module. Also, the bounding box for any of the modules defined by l and s needs to be contained within the device boundary. Given the above constraints $|P|$ is $O(|M|.n^4)$, where n is *max(Number of columns in device , Number of rows in device)*.

Given a floorplan representation and a suitable cost function the FPGA floorplanning problem could be solved through stochastic methods. However, we encourage other researchers to study this problem more carefully and research better algorithms.

Problem 5.1: Let the resource type of a particular module, v, be specified by t(v). The floorplan, F, will seek to optimize the cost function C:

$$C = \sum \alpha |ratio\text{-}1| + \beta(external\text{-}wire\text{-}length) + \gamma(overlap)$$

Where *ratio* is the aspect ratio of the module, *external-wire-length* is the total wirelength measured as center-to-center manhattan distance between modules times the number of nets between module pairs, and *overlap* is the total overlap between pairs of module rectangle boundaries. α, β, and γ are weights used to trade off different cost criteria.

6. PLACEMENT

Placement forms the back bone of any good physical synthesis system. Placement is a well researched problem in the standard cell domain. However, traditional ASIC scalable placement techniques such as min-cut or force-directed based algorithms are effective when the layout space is geometric (e.g., a mesh).

As presented earlier, present FPGAs are departing from geometric layouts, due to the existence of "express" (e.g. double/hex/long) nets in the routing architecture. Express wires in the FPGA fabric create extra edges in the mesh (that is usually used to model standard cell designs) and route structures that hop across the fabric remove edges from the mesh. In other words, in recent FPGAs the re-configurable logic blocks that are physically distant, can be connected to each other with almost insignificant delay penalty.

It follows that direct adaptation of the traditional ASIC placement techniques simply fail to produce high quality results for FPGAs. FPGA placement techniques have to be *layout-aware* in order to capture the change in the topology of the placement surface. We will take a closer look at placement in the context of today's FPGAs, and introduce a few fundamental problems that require extensive research to be reasonably addressed.

Problem 6.1: What is a good cost function for an FPGA placement tool?

In ASIC domain, the length of a wire is proportional to its delay. As we will elaborately discuss in section 7, that is not the case for designs implemented on an FPGA. Therefore, traditional standard cell placement cost functions are not directly applicable to FPGA domain, due to fundamentally different routing model of the FPGAs. Apart from the classic wirelength cost function, new timing driven cost functions are also needed, because traditional ASIC critical path definitions (physically longest path) would be different in FPGA domain.

Problem 6.2: How can an FPGA placement tool estimate congestion and routability?

In standard cell designs, quick and fairly accurate congestion estimation of a given design can be performed by calculating the density of the nets. The density (the chromatic number of the interval graph corresponding to nets in a channel) represents the maximum number of nets that have to be put next to each other in that particular routing channel. While density serves as a fairly reasonable metric in standard cell domain, it is no longer effective for FPGA designs, the reason being the underlying segmented routing architecture of the FPGAs. For example, a short local net can be realized using a long net that passes through the entire channel, which might translate into a congested area in *another* location of the channel.

6.1 Experimental Setup

We have built a placement tool based on Dragon [24]. Dragon is a state-of-the-art min-cut based standard cell placement tool, which uses simulated annealing to optimize the placement in each partition. Our new placement tool performs global placement of circuit netlists on a simplified FPGA layout model.

In our simplified model, called M^i, each node of the grid is directly connected to all of its neighbors of distance less than or equal to i in all four directions (Figure 6.1). We assume that the delay of all nets is equal in M^i. Hence, the delay associated with a set of nets connecting two nodes is proportional to the number of

net segments (hops) that are used to connect the two nodes. It follows that the traditional total wirelength metric would translate into the total number of the net segments (hops) that are used to connect pins. In [27] authors have shown that this metric correlates very well with the post routing delay.

Our global placement tool uses the total number of net segments as its cost function. Furthermore, we ignore the constraint of limited number of wire segments at the global placement stage. Therefore, we always assume that a net can be implemented using the fastest (least number of segments) possible way at this stage.

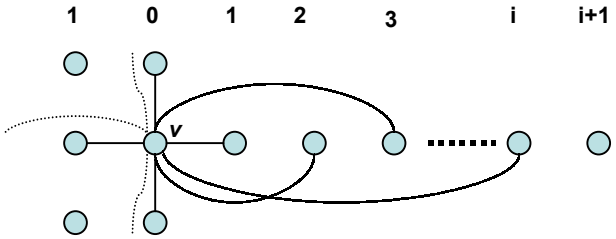


Figure 6.1. In Model M^i each node is connected to all neighbors that are at most i units distant (in all four directions). Connections to node *v* only on one of the four directions have been shown.

Interestingly M^i can model both mesh, and FPGA layouts. If i=1, then each node of the grid is only connected to its immediate neighbors. That is M^1 is the well-known mesh layout, which is normally used by traditional ASIC physical design tools. Larger values of i can be used to capture the advances of the technology that allow more long nets in the FPGA routing architectures.

We have implemented two global placers based on Dragon. The first version places the CLBs by only min-cut partitioning of the given netlist, while the second version tries to improve the results in each partition using simulated annealing. The cost function for both of the placers is number of net segments in M^i.

Circuit	# of cells	# of nets
ibm01	12,028	11,753
ibm02	19,062	18,688
ibm07	44,811	44,681
ibm08	50,672	48,230
ibm09	51,382	50,678
ibm10	66,762	64,971
ibm11	68,046	67,422
ibm12	68,735	68,376

Table 6.1. Details of placement benchmarks

We have created a set of synthetic benchmarks that have the same number of nodes and connectivity as some IBM benchmarks. However, the nodes in our benchmarks are assumed to have similar sizes. This is required because nodes correspond to FPGA CLBs that are similar, unlike different standard cell sizes in ASIC domain. Table 6.1 demonstrates the original circuits that were used to create our testbenches and their characteristics. The circuits are part of IBM benchmark suit that are designed for standard cell placement.

6.2 Experimental Results

We have placed the benchmarks shown in Table 6.1 using both of our placement tools. The total number of wire segments for each placed benchmark in M^i is reported in Tables 6.2 and 6.3. The unit for each number is the grid size in M^i, which is constant for all experiments.

Circuit	i = 1	i = 2	i = 3	i = 4	i = 8
ibm01	122.1	67.2	48.9	38.8	25.9
ibm02	334.4	176.1	126.3	94.9	58.5
ibm07	777.8	411.8	292.2	234.6	143.0
ibm08	891.9	479.0	328.3	264.9	153.2
ibm09	738.6	399.0	284.4	221.2	136.0
ibm10	1177.7	608.5	439.8	337.7	201.3
ibm11	1111.4	582.4	417.7	325.3	198.7
ibm12	1562.3	850.7	565.8	443.2	260.9
Average	839.5	446.8	312.9	245.1	147.2

Table 6.2. Number of net segments (in thousands) on M^i, using a partitioning-only placement tool.

Table 6.2 illustrates the results for the first version (placement using only partitioning) of our placement tool, while Table 6.3 exhibit the results for the second version of the placer (placement using partitioning + simulated annealing). The numbers shown in both tables are in million wire segments. The last row of both tables indicates the average wire segments over all benchmarks.

Circuit	i = 1	i = 2	i = 3	i = 4	i = 8
ibm01	97.7	50.9	38.5	33.0	24.6
ibm02	265.8	141.6	102.4	82.8	52.7
ibm07	602.0	316.2	240.9	188.3	124.0
ibm08	650.1	361.0	254.8	208.0	136.5
ibm09	575.0	307.8	232.9	183.4	123.8
ibm10	935.6	499.0	377.2	289.2	190.2
ibm11	843.0	448.6	338.9	264.9	179.9
ibm12	1225.0	623.8	443.2	364.4	227.9
Average	649.3	343.6	253.6	201.7	132.4

Table 6.3. Number of net segments (in thousands) on M^i, using a partitioning+annealing placement tool.

The chart in Figure 6.2 demonstrates the variations of the average number of wire segments (last row of Tables 6.2 and 6.3) over different layout models. Larger values of i correspond to newer generations of the FPGAs, where longer wire segments, and more number of wire segments exist in the routing architectures. Naturally, the number of wire segments required to connect pins, decreases by addition of extra nets into layout.

As can be observing from Figure 6.2, Incrementing *i* from 1 to 2 almost halves the number of net segments required to route the connections. It can be inferred that there are many net segments whose length is greater than 1 in at least one of the two X or Y directions. All such nets would be routed with less number of net segments in M^2 compared to M^1. Note that nets of length 1 (in either of the two X and Y directions) incur the same cost in all M^i.

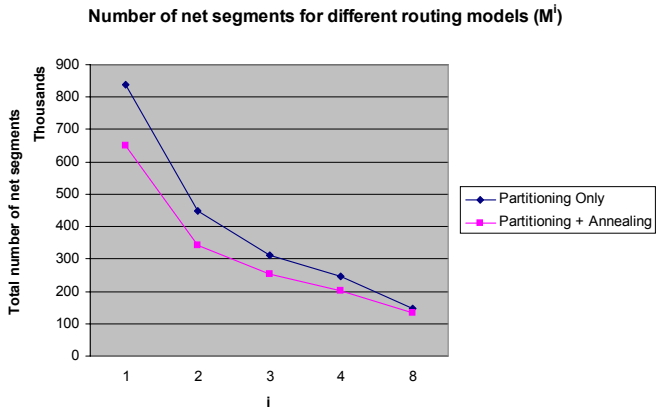


Figure 6.2. Average number of wire segments for different layouts.

On the other hand, most of the nets in a placed design are local nets whose lengths are not very large. Therefore, excessive increasing of *i* does not exhibit the same trend in reduction of the cost function. For example, the number of required net segments in M8 is only about %20 of M^1, where about %50 of savings happen when we move from M^1 to M^2. Hence, our study provides a rudimentary measure for evaluating area-delay tradeoffs involved in designing routing architectures for FPGAs.

7. DELAY ESTIMATION

Recent works have tried to estimate wire delays at various levels of design stages [4], [5], [11], [12]. Almost all of these previous works have exploited features (routing resources, gridded architecture etc.) of targeted FPGA architecture to estimate delays. Though relevant to the targeted FPGA, most of these previous methodologies cannot be applied to recent multi-million gate FPGAs because of the complexity of the estimation process [11][12].

Terminologies and Experimental Setup:

From now on whenever we say *distance* we mean Manhattan distance between the CLBs in which driver and driven pins are respectively located. Similarly, *delay* would mean delay between driver and driven pins.

We will show experiments on one representative industry design, ind_com (similar results were observed on other large designs). This design has 8.7M gates and is targeted for 2v8000 VirtexII FGPA device with 112x104 CLBs. All the data for these experiments was generated using Xilinx's Place and Route tool, PAR [25]. PAR was run in high-effort mode for timing and routing optimization.

Variables affecting wire delays:

Some of the variables which traditionally have been explored for wire delay estimation are: fanout of the net (connecting driver-driven pin pair), distance between driver-driven pin pair and routing congestion [12]. Of these, routing congestion is the hardest to measure and depends heavily on routing algorithm being used. Congestion estimation is beyond the scope of this work. We will limit ourselves to studying the impact of net fanout and distance between driver-driven pin pair on wire delays.

Fanout of the net connecting pin pair

Fanout of the net has been shown to correlate really well with the delay in Standard-Cell design methodologies and has been used extensively to derive wire-load models for net delays [3]. However, our experiments show that in the FPGA domain a very weak correlation exists between the two. In Figure 7.1, we show post-routing delay for pin pair (with a distance of 2) versus net fanout plot for ind_com design. For different values of the fanout the range of delays is almost same, giving rise to the notion that fanout has very little impact on delay. The reason why fanout of the net does not impact delay lies more in the routing architecture of the Xilinx FPGA devices. These devices use buffered inter-connects to route the nets [25]. Routing switches break the net at regular intervals and hence the traditional fanout based wire-load models, using *Elmore delay* [7], cease to work.

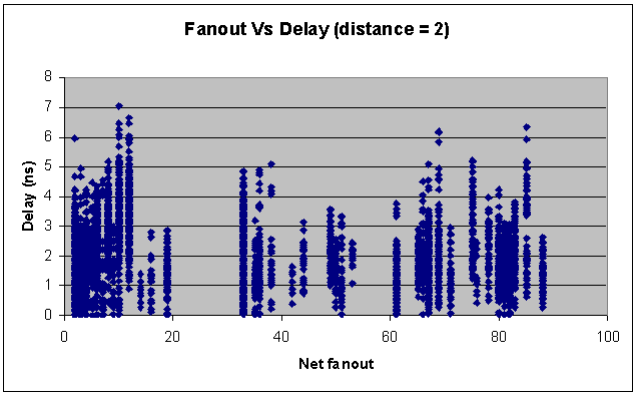


Figure 7.1. Net fanout Vs delay for a fixed distance.

Distance between pin pairs

Authors in [12] have shown that delay between a pin pair has no direct linear relation-ship with the distance between them. To corroborate their observation, for every driver-driven pin pair (for nets with fanout of 2) in ind_com design we plot distance versus post-routing delay. It is evident from the plot, Figure 7.2 that even though the delay seems to be increasing with the distance, for a given distance the range of delays is too large. Trying to fit a

linear line to extrapolate the delay for a given distance will have huge error margin.

However, if we plot delays for a given distance, interesting patterns emerge. Figure 7.3 shows the plots of delays of all the driver-driven pin pairs in ind_com design with a distance of 2. We see that most of the delays are centered on very few vertical lines thereby indicating that delays are combination of discrete values.

To motivate our argument for a discrete delay model, it would help to take a look at the routing architecture in Figure 2.2. It is this discrete routing structure which gives rise to vertical lines in Figure 7.3. For a distance of 2 CLBs between a driver-driven pair, the number of possible routes is limited. Such a distance can be routed using either of the following: one double line, two double lines, or a direct line and a double line etc. The delays for these different types of lines are almost constant (minor variations might occur due to switch-box delays) and have no relation to each other. For example, delay of a hex line is not three times the delay of a double line or six times the delay of a direct line but is only slightly larger than the delay of a double or direct line.

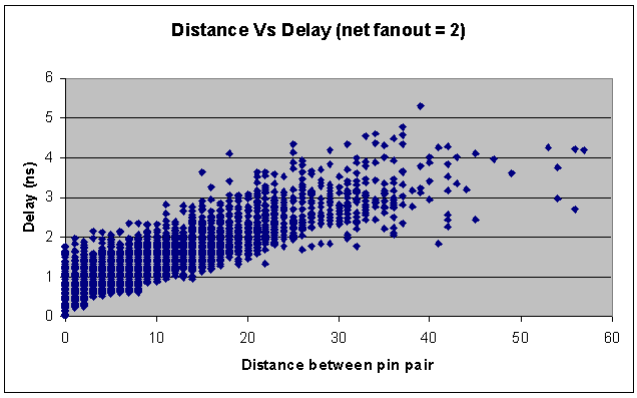


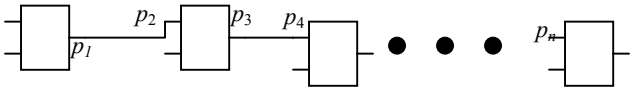
Figure 7.2. Distance Vs delay for a fixed fanout.

Based on the plots in 7.1 and 7.2 we see that traditional delay estimation techniques that depend on the fanout of the net in question and the distance between specific pin pairs fall apart in the FPGA domain. Wang et al [27] illustrate an algorithm to estimate the delays between pin pairs for VirtexII architectures. The algorithm is based on estimating the type of routes used between the two CLBs on the FPGA fabric. In [27] the design is already placed and routing has not been performed. Accurate delay estimations, however, also need to be performed earlier in the design flow to enable performance based logic and placement optimizations.

Problem 7.1: Let $d(p_i, p_j)$ be the delay associated with the net segment between source pin p_i and the sink pin p_j. Predict the delay, $e(p_i, p_j)$ prior to placement to minimize E, where:

$$E = \Sigma |d(p_i, p_j)\text{-}e(p_i, p_j)|$$

A more relevant problem formulation would be the one that reduces the error in delay estimates on all timing paths found in the design.



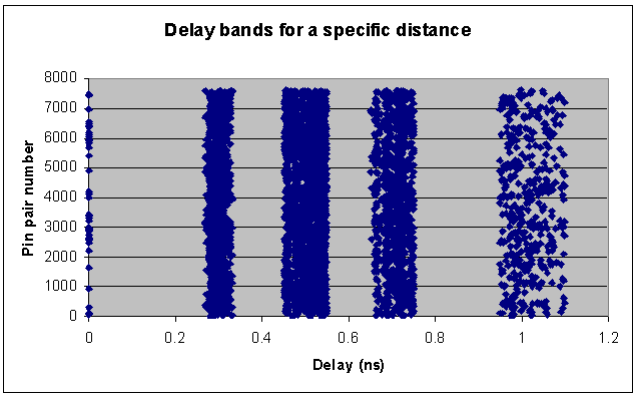


Figure 7.3. Discrete delay bands for a distance of 2.

Problem 7.2: Let P_m^n be the m^{th} path in the design with n timing pins along the path. Let $d(p_i, p_{i+1})$ be the delay associated with the net segment between source pin p_i and the sink pin p_{i+1} along the path. Predict the delay, $e(p_i, p_{i+1})$ prior to placement to minimize E, where:

$$E = \Sigma_{\forall m} |\Sigma_{\forall i \, s.t \, (i \le n) \, and \, (i \bmod 2 = 1)} \, d(p_i, p_{i+1})\text{-}e(p_i, p_{i+1})|$$

8. CONCLUSION

Advances in technology have lead to creation of high capacity and heterogeneous FPGA devices. Many underlying ASIC domain assumptions, such as continuous area or customized routing segments, do not hold for such architectures. Therefore, a direct retargeting of traditional physical design techniques does not provide quality results for reconfigurable devices. It follows that many well-researched physical design problems need to be reformulated and investigated in order to fully utilize modern FPGA devices.

In this paper, we presented a number of basic physical design problems, and showed why traditional techniques would fail to produce reasonable results. Particularly, we overviewed traditional partitioning, floorplanning, placement, and delay estimation techniques, and highlighted their shortcomings for producing good results for modern FPGAs.

In each section, we formulated a few underlying problems that can potentially initiate fundamental research efforts in that area. Moreover, we proposed a simplified yet powerful model for representing segmented routing architectures. The model can be utilized in many different physical deign stages such as placement, routing and delay estimation. We encourage researchers to investigate the proposed ideas and problems further, in order to develop novel and efficient FPGA physical design tools.

9. ACKNOWLEDGEMENTS

The authors would like to thank Mr. Bo-Kyung Choi for his generous help in implementing the placement experiments, and fruitful discussions.

10. REFERENCES

[1] A. E. Caldwell, A. B. Kahng, S. Mantik, I. L. Markov and A. Zelikovsky, "On Wirelength Estimations for Row-based Placement," ISPD, pp. 4-11, 1998.

[2] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Can Recursive Bisection Alone Produce Routable Placements?" in Design Automation Conference, pp. 477-482, 2000

[3] C. Chen and C. Tsui, "Timing Optimization of Logic Network using Gate Duplication", ASP-DAC, pp. 233-236, 1999.

[4] C. S. Chen, Y.-W. Tsay, T. HwangA. C. H. Wu and Y.-L. Lin, "Combining Technology Mapping and Placement for Delay-Optimization in FPGA Designs," ICCAD, pp. 240-247, 1993.

[5] J. Cong and Y. Ding, "An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," ICCAD, pp. 48-53, 1992.

[6] W. E. Donath, "Placement and average interconnection lengths of computer logic", IEEE Transactions on Circuits and Systems, CAS-26(4):272-277, April 1979.

[7] W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wide Band Amplifiers," J. Applied Physics, 19(1), 1948.

[8] J. M. Emmert, and D. Bhatia, "A Methodology for Fast FPGA Floorplanning", Proc. FPGA, 1999.

[9] P. N. Guo, C.-K. Cheng, and T. Yoshimura, "An O-Tree Representation of Non-Slicing Floorplan and Its Applications", Proc. DAC, pp. 268-273, 1999.

[10] M. Z. Kang and W. Dai., "Arbitrary Rectilinear Block Packing Based on Sequence Pair", Proc. ICCAD, pp. 259-266, 1998

[11] T. Karnik, "Hierarchical Timing-Driven Partitioning and Placement for Symmetrical FPGAs," PhD thesis, University of Illinois at Urbana-Champaign, 1995.

[12] T. Karnik and S. M. Kang, "An Empirical Model For Accurate Estimation of Routing Delay in FPGAs," ICCAD, pp. 328-331, 1995.

[13] G. Karypis and V. Kumar, "Multilevel k-way Hyper-graph Partitioning", in Design Automation Conference, pp. 343-348, 1999

[14] J. M. Kleinhans, G. Sigl, F. M. Johannes and K. J. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization", IEEE Transactions on Computer Aided Design, 10(3): 365-370, 1991

[15] H. Murata, K. Fujiyoshi, S. Nakatake and Y. Kajitani, "VLSI Module Placement Based on Rectangle-Packing by the Sequence Pair", IEEE Trans. On CAD, vol 15(12), pp. 1518-1524, 1996

[16] R. Nair, C. L. Berman, P. Hauge, E. Yoffa, "Generation of Performance Constraints for Layout", IEEE Trans. On CAD, vol. 8(8), pp. 860-874, 1989.

[17] R. H. J. M. Otten, "Efficient Floorplan Optimization", ICCD, pp. 499-503, IEEE/ACM, 1983

[18] R. H. J. M. Otten, "Automatic Floorplan Design", Proc. DAC, pp.261-267, 1992

[19] L. Stockmeyer, "Optimal Orientation of Cells in Slicing Floorplan Designs", Information and Control,57(2), pp. 91-101, 1983

[20] X. Tang, R. Tian and D. F. Wong, "Fast Evaluation of Sequence Pair in Block Placement by Longest Common Subsequence Computation", DATE 2000, pp. 106-111.

[21] X. Tang and D. F. Wong, "FAST-SP: A Fast Algorithm for Block Placement Based on Sequence Pair", ASPDAC 2001.

[22] A. Ranjan, K. Bazargan, M. Sarrafzadeh, "Fast Hierarchical Floorplanning with Congestion and Timing Control", IEEE International Conference on Computer Design (ICCD), pp. 357-362, September 2000.

[23] J. Vygen, "Algorithms For Large-scale Flat Placement", Proc. Design Automation Conference, pp. 746-51, 1997.

[24] M. Wang, X. Yang and M. Sarrafzadeh, "Dragon2000: Standard-Cell Placement Tool For Large Industry Circuits", ICCAD, pp. 160-163, 2000.

[25] Xilinx Inc., http://www.xilinx.com/

[26] N. Selvakkumaran, A. Ranjan, S. Raje, G. Karypis, "Partitioning Algorithms for FPGAs with Heterogenous Resources", Symposium on Field Programmable Gate Arrays, 2004.

[27] M. Wang, A. Ranjan, S. Raje, "Multi-Million Gate FPGA Physical Design Challenges", ICCAD, pp. 891-898, 2004.