Interconnect Resource-Aware Placement for Hierarchical FPGAs*

Amit Singh, Ganapathy Parthasarathy, Malgorzata Marek-Sadowska
Department of Electrical and Computer Engineering
University of California, Santa Barbara, Santa Barbara, CA 93106, USA
{asingh@cornet.ece, gpartha@bigbend.ece, mms@ece}.ucsb.edu

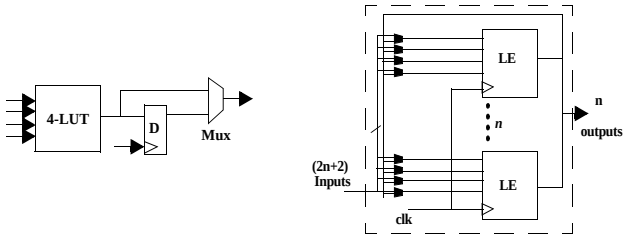
Abstract

In this paper, we utilize Rent's rule as an empirical measure for efficient clustering and placement of circuits on hierarchical FPGAs. We show that careful matching of design complexity and architecture resources of hierarchical FPGAs can have a positive impact on the overall device area. We propose a circuit placement algorithm based on Rent's parameter and show that our clustering and placement techniques can improve the overall device routing area by as much as 21% for the same array size, when compared to a state-of-art FPGA placement and routing tool.

1. Introduction

Field Programmable Gate Arrays (FPGAs) have gained rapid commercial acceptance because their user-programmability offers instant manufacturing turnaround and low costs. However, FPGAs are constantly hard pressed to keep up with the requirements of the more complex and larger scale circuits which are being targeted for them. Most FPGA vendors are using the advances in DSM technologies to ship devices that can implement million gate designs. Such high logic density comes at an enormous interconnect cost since most of the device area is usually devoted to interconnects. A goal to achieve close to 100% logic utilization in these million gate devices proves extremely expensive in terms of area and power.

Speed and area-efficiency of an FPGA are directly related to the granularity of its logic block. For a device consisting of very fine grained logic blocks, many connections must be routed between the different blocks. This results in complicated routing and long routing delays. Coarse-grained blocks, on the other hand, are very area-inefficient and have long delays. Recently, FPGA vendors have introduced hierarchical FPGAs consisting of logic clusters. Examples of such devices are the Xilinx Virtex [21] and the Apex 10K [22] from Altera. In these architectures, groups of Logic Elements (LEs) are clustered to provide better performance, specially for communication oriented designs. Figure 1.a shows the structure of an LE built of a 4-input look-up table (LUT) and a Flip-Flop (FF). Each logic cluster consists of several of these LEs as shown in Figure 1.b. Dedicated local routing is provided inside each cluster for communication between the local LEs.



a) A Logic Element (LE) **b) A logic cluster of size n**

Figure 1: Logic Element and Logic Cluster

Prior research [3][17] on hierarchical FPGA architectures has focussed mainly in the area-delay trade-off stemming from the size and structure of the clusters. Betz et al. [3][17] presented two clustering algorithms, *VPack* and a timing-driven version, *T-VPack*. A recent work, *RPack*[2], presented a routability-driven packing algorithm which first identifies routability factors, prioritizes these factors into the clustering cost function and achieves fewer routing tracks than *VPack*[3]. This approach, however, produces routing track counts comparable to those generated by *T-VPack*[5][17].

In this paper, we explore the routability of clustered (from now on, we use the terms cluster and hierarchical interchangeably) architectures. We utilize Rent's rule [11][15] as a complexity metric for improving the clustering and placement of circuits on hierarchical FPGA architectures, such that the overall device utilization is improved. For MCNC benchmark circuits placed and routed on hierarchical FPGAs, an improvement of as much as 21% in terms of overall area utilization is achieved in comparison to results obtained from a state-of-art FPGA tool [3][17].

The rest of the paper is organized as follows. Section 2 highlights the previous work done in the area of FPGA routability analysis and gives an overview of Rent's rule. Section 3 presents the problem statement. Section 4 shows the interconnect resource driven bottom-up clustering technique which is built on top of *T-VPack* [17]. The next section discusses the Rent's rule based FPGA placement. This is followed by experimental results in Section 6 and their analysis. Section 7 concludes this paper.

2. Previous Work and Rent's Rule

This sections summarizes previous work done to capture interconnect complexity in FPGAs. This is followed by an overview of Rent's rule.

2.1 Previous Work

Researchers have tried to capture interconnect requirements for circuit routability using a variety of metrics. El Gammal [12] used a stochastic model to estimate routability of channeled gate arrays. This was extended by Chan et al. [7] to FPGAs. Alexander [1] used a sequence of Steiner tree-on-a-graph approximations to determine routing solutions for FPGAs. Wu [20] showed that simplified variants of FPGA routing are reducible to the graph coloring problem and used this result to show the NP-hard nature of FPGA routing. Wood [19] estimated FPGA routability using boolean satisfiability and Binary Decision Diagrams (BDDs), extending the work that Devadas [10] had done for ASIC routing.

Most routability methods are very time consuming and difficult to use directly as a placement cost function. They also do not capture the architectural issues that are typical to FPGAs, like the length of segments used, level of clustering and complexity variations in different

* This work was supported in part by the GSRC DARPA-MARCO grant and in part by California MICRO program through Xilinx.

parts of the designs to be implemented. In this work, we look at empirical methods for routability estimation. The metric we use is based on Rent's rule. Rent's rule was first proposed by Rent at IBM and also derived by several others including Donath [11], Brown [6] and Landman [15]. Since then, Rent's rule has been used extensively as a predictor of circuit complexity. Hagen et al. [13] used the spectra-based ratio cut partitioning to obtain partitioning trees with the lowest observed Rent's parameter. They argued that such partitioning trees correspond to area-efficient top-down layouts. Van Marck et al. [16] used circuit interconnect behavior to study Rent's exponent of circuits and proposed a method to capture the variation in complexities in different parts of designs.

DeHon [9] showed that for hierarchical FPGAs, 100% logic utilization is not necessarily beneficial for overall device area minimization. He presented some initial evidence to support this claim and presented a technique for depopulating gates in a hierarchical array. His results indicate that a careful partitioning of designs and depopulation of logic clusters can result in better FPGA resource utilization. This is an extremely important consideration in the era of Deep Submicron (DSM) technologies where FPGA vendors claim million gate FPGAs and ship devices in which most of the silicon area (as much as 85-90%) is devoted to interconnects. By intelligently under-utilizing logic resources of an FPGA using Rent's rule, we can improve both placement and routability in the device, thereby saving area and reducing die sizes. We show the inherent advantage of this empirical metric, both during the logic clustering, and the placement phases. This technique is specially effective for hierarchical FPGA architectures where different levels of hierarchy may have different routing resource complexities.

We use a combination of empirical analysis and analytical methods to study different circuits, perform netlist clustering (with modification in cases when netlist have non-uniform local Rent's parameter) and achieve efficient place and route solutions for hierarchical FPGA architectures.

2.2. Rent's Rule Overview

In 1960, E.F. Rent of IBM published two internal memoranda that contained the log plots of "number of pins" versus "number of circuits" in a logic design. Such plots tend to form straight lines in a log-log scale and follow the relationship:

$$N_{io} = KB^{p} \qquad (1)$$

where N_{io} = number of pins in a module, B is the number of blocks in the module, K is the number of connections per block in the module, and p is a parameter known as *Rent's parameter*. Rent used this empirical measure to estimate routability of logic circuits.

In 1995 Van Marck et al. [16] showed that for a placed circuit that minimizes the total wire-length, the *local* Rent's parameter of a design can be determined from the slope of the log-log plot of *number of interconnections from a block* versus *length of interconnection*. For this placed design, he calculated the *local* Rent's parameter of the design, as follows:

For a given block, v_i, in a design, count the number of interconnections, N_j, of length l_j, from block v_i (i.e. both input and output nets), and find the best-fitting line on the log(N_j) versus log(l_j) plot. The *local* Rent's parameter of this block is then defined as the slope of this log-log plot. Their method, essentially, proposes the use of net length distribution as a means to calculate the relation between the interconnect complexity and block count. This is useful for designs in which different parts can have widely different complexities.

3. Definitions and Problem Statement

We first present our hierarchical architecture model and define key terms. This is followed by the formal problem statement.

3.1 Architecture Model

Figure 2 shows a typical hierarchical FPGA architecture. Each logic cluster (CLB) in this figure is similar to the one shown in Figure 1.b and consists of n LEs. Local routing is provided inside the cluster, which allows all the n LEs to connect to each other through the use of multiplexers. Routing between the clusters is through inter-cluster tracks. The number of tracks between any two neighboring clusters is uniform and is called the channel width. The number of logic clusters that each wire-segment spans before going through a switch box is called the track segment length. All switch-boxes are of the *subset* type and provide inter-cluster routing from any track i to its adjoining horizontal or vertical segment i. Switches are a 50-50 mix of tri-state buffers and pass-transistors. All pins on a cluster can connect to any of the available tracks in its adjacent channels. We assume that a cluster of size n has $(2n+2)$ input pins and n output pins. Indeed, this is sufficient to achieve full logic connectivity as shown by Betz in [5]. In addition, we assume that all segments are of lengths 1. Even though this architectural decision influences our results, we believe that the tendency shown in these results will hold for similar architectures with different parameters and segment lengths.

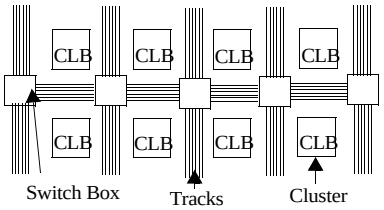


Figure 2: General Architecture model

3.2 Definition of Key Terms

The *local* Rent's parameter of a design, P_d, is determined from the log-log relationship between the number of used pins on a cluster versus the number of used LEs in the cluster. We determine the FPGA architecture's Rent's exponent P_a from the log-log relationship between the number of available cluster pins versus the cluster size. P_a captures the interconnect resource growth of an FPGA [9]. *Network utilization* is defined as the ratio of used interconnect to the available interconnect in a FPGA fabric in which the minimum number of tracks is the required channel density.

The *global* Rent's parameter of the design, is the value of P_d at the top-most level in the design hierarchy. However, for random designs, this may not be the same as the *local* Rent's parameter of the design at some level in the design hierarchy. For designs that are already placed, Van Marck [16] proposed a method for calculating the *local* Rent's parameter. This method is discussed in Section 2.2.

Since the FPGA device has uniform interconnect resources, P_a at the local level is a fairly accurate measure of the overall device P_a. To check the accuracy of the P_a value calculated using formula (1), we placed and routed uniform complexity benchmarks circuits having values of P_d ranging from 0.2 to 0.9. The synthetic benchmarks were created using ***gnl*** [18] by doing a bottom-up clustering of blocks having a pre-defined Rent's parameter, such that a block in a higher level has a

P_d value similar to the P_d values in the lower levels. For each generated benchmark, the routing resource utilization was measured. Figure 3 shows a plotted graph of the routing utilization versus P_d for a simple island style architecture with segmentation of 1. Similar experiments

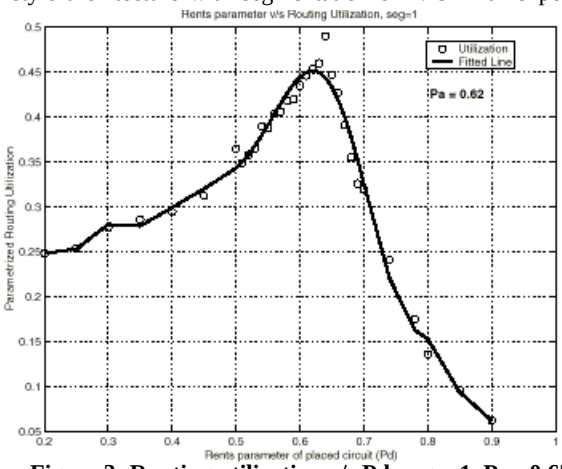


Figure 3: Routing utilization v/s Pd, seg = 1, Pa =0.62

are performed on clustered architectures. We obtain P_a by drawing the best-fit line and using the value that gives the highest network utilization for the same architecture.

We now give a formal problem definition:

Problem Statement: *Given a hierarchical 2-D mesh FPGA chip with Rent's exponent, P_a, and a design mapped to k-LUTs, having Rent's exponent P_d, fit the design in the chip so that the area of the mapped, placed and routed design is minimized, subject to the constraint of a given bounding box aspect-ratio.*

4. Logic Clustering using Rent's rule

Most modern day designs exhibit Rent's parameter of $p>0.5$. Since designs are usually hierarchical, with different hierarchies having different interconnect complexities and even different modules (blocks) in the same hierarchy having different interconnect complexity, a trial and error clustering and placement implementation method may lead to unnecessary congestion and too much unused routing resources.

Our method, aims to alleviate this problem by depopulating logic clusters. This bottom-up clustering has two phases. The first phase finds the upper bound on the cluster pin size based on the architecture's Rent's parameter. This allowable number of cluster pins is bounded by the cluster size and the maximum available pins in each cluster in the architecture. That is, if j is the number of pins which can be used, then $k+1 \le j \le 3n+2$, since the minimum number of pins is $k+1$ (k is the LE size) and the maximum number of pins is $2n+2+n = (3n+2)$. We can calculate j as shown below.

$$j = (k+1)n^{Pa} \qquad k+1 \le j < 3n+2 \qquad (2)$$

where $(k+1)$ is the average number of connection per k-input LE (this value results from the technology mapping phase), n is the cluster size, and P_a is the architecture's Rent's parameter as found from the synthetic benchmark circuits. The value of j constraints the local Rent's parameter of any logic cluster in the design, P_d, to be no greater than P_a.

The second phase performs clustering based on the new set of block pin constraints. Clustering is performed in a bottom-up fashion and is based on an LE attractiveness gain function described in [17]. Each LE in the unclustered design has a gain value associated with it. This gain value is a function of 2 parameters, (i) an attractiveness function, which measures the attraction that the unclustered LE, B, has to the current cluster C, $A(B) = |Nets(B) \cap Nets(C)|$, and ($ii$), a Critically function, $Cr(B)$, which aims to cluster those unclustered LEs which are the most timing critical. The gain function is defined as :

$$Attraction(B) = \alpha \bullet Cr(B) + (1-\alpha) \bullet \frac{A(B)}{G} \qquad (3)$$

The default values of α and G are 0.75 and j respectively. We perform clustering one LE at a time starting with the unclustered LE with the highest gain value (this is the LE with the highest criticality as found by doing a static timing analysis on the network of LEs). After the LE has been absorbed into a cluster, the gain values for the remaining unclustered LEs are recomputed using formula (3) and the unclustered LE with the highest gain is chosen. This LE can be added to the current cluster only if the capacity, and cluster pin constraints (as found in phase 1) are satisfied. In the case when either of these 2 constraints are violated, we form a new cluster and absorb the unclustered LE into the new cluster. Once all unclustered LEs have been absorbed, the original network of unclustered LEs has been transformed into a clustered network of LEs.

Our clustering technique guarantees a fair amount of uniformity in the clustered design's complexity as measured by Rent's parameter and gives our placement tool (described in the next section), a fairly uniform depopulated hierarchical circuit where the worst case P_d of any logic cluster is equal to P_a. The placement tool then recognizes the variations in the *spatial* complexities of the clusters and generates placement solutions which result in fewer routing tracks and shorter wirelengths after routing.

5. Placement using Rent's rule

We use a simulated annealing [14] place and route tool built on top of a state-of-art FPGA place and route tool *VPRv4.30* [3]. The placement cost function is:

$$\sum_{i=1}^{Nets} q(i)\left(1 + \sum_{\forall(k \in i)} \left|P_{dk} - Pa\right|\right)(wirelength_i) \qquad (4)$$

In the placement cost function, P_{dk} corresponds to the *local placed* Rent's parameter of net i and the clusters connected to it, as found using Van Marck's technique. This local placed Rent's parameter is calculated for each net in the clustered netlist. For each net, we consider the placement of the clusters connected to the net, and calculate the interconnect length requirements due to the current placement. In a sense, if we consider the individual clusters as constituting the lower level of the design hierarchy, then the clusters and their surrounding interconnects form the higher level of the design hierarchy. Therefore, in the placement phase, we are not concerned with the Rent's parameters of the individual clusters but rather, with the Rent's parameter of the clustered netlist at the higher level of hierarchy. P_{dk} calculation takes $O(nlog(n))$ time, since all the net-lengths need to be sorted. At each stage of the placement algorithm, any cluster swap results in an updated local placed Rent's parameter for only those clusters and the nets connected to them. Wirelength is calculated as the minimum perimeter (bounding-box) that encloses a net and the cost function minimizes the scaled sum of these bounding boxes over all nets. Each bounding box is scaled by 2 parameters. $q(i)$ is a factor similar to the one used in [3] and ranges from 1 for nets with few pins to 2.79 for nets with 50 terminals. The 2nd parame-

ter, $(1 + \sum |P_{d_k} - P_a|)$, attempts to minimize the sum of differences between P_a and the local placed P_d of all clusters on a net. In cases where local placed P_d is greater than P_a, the scaling has the effect of moving congested blocks into sparsely populated areas of the fabric.

6. Experimental Results

We have implemented our bottom-up clustering and placement technique, *i*RAP (Interconnect Resource Aware Placement), on top of the state-of-art FPGA place and route tool, VPR (v4.30) [3]. We experimented with 2 kinds of benchmarks *(i)* synthetic benchmarks [18] with fixed design Rent's parameter and *(ii)* random MCNC benchmarks. Each design was mapped into 4-input LUTs using flowmap [8] and clustered using the technique outlined in section 4. Mapped and clustered designs were then placed and routed using the technique outlined in section 5. Tables 1 and 2 show the placed and routed results for 11 MCNC benchmark circuits for cluster sizes of 4 and 8 respectively. Figures 4-6 show the benchmark circuit C432 when clustered, placed and routed under 3 different conditions. Figure 4 shows the VPRv4.30 [3] generated packing (packed using *T-VPack* [17]) and routing for this circuit. Each block in the fabric is a cluster of size 4 and contains 10 input and 4 output pins. The placed and routed circuit utilizes 13 tracks with 5 empty clusters. Figure 5 shows the same place and route tool result for the circuit which has been depopulated to meet the architecture's Rent's parameter. The track reduction in this case is 15% without any logic area and critical delay overhead (most empty clusters were simply filled up). Figure 6 shows the place and route result for the depopulated circuit using *i*RAP. The number of tracks needed to route this circuit is 10 in this case, or a further improvement of 9% in number of tracks over the one in Figure 5, without any logic area overhead.

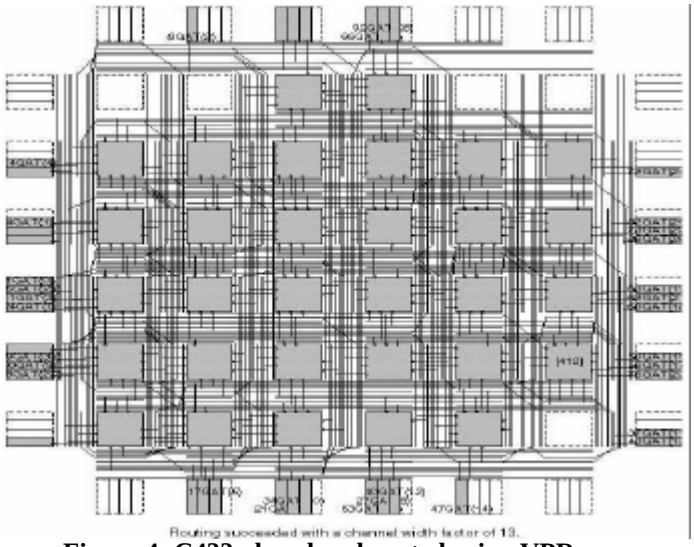


Figure 4: C432 placed and routed using VPR

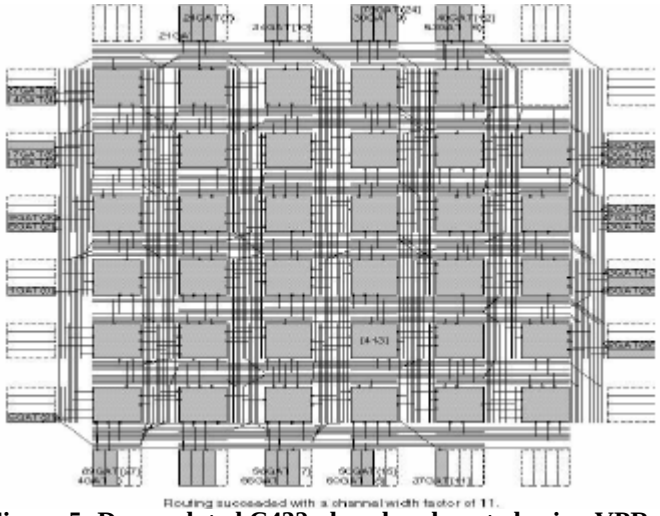


Figure 5: Depopulated C432 placed and routed using VPR

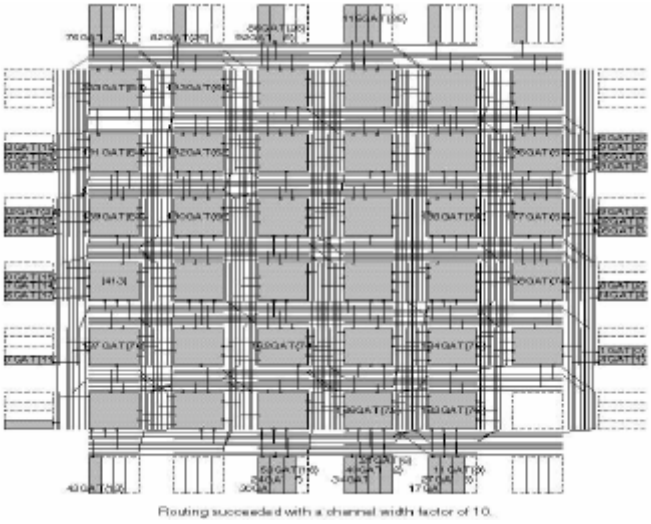


Figure 6: C432 placed and routed using *i*RAP

Table 1: Cluster size 4

Circuit	T-VPack+VPR				VPR+depopulation				*i*RAP			
	nx,ny	Channel	WL	Trans.	nx,ny	Channel	WL	Trans.	nx,ny	Channel	WL	Trans.
C432	6,6	13	904	2739.90	6,6	11	819	2400.54	6,6	10	753	2248.21
C499	5,5	16	806	3432.08	5,5	13	706	2880.38	5,5	12	664	2719.55
C880	7,7	16	1510	3142.41	7,7	13	1292	2636.17	7,7	13	1243	2636.17
C2670	24,24	18	4327	2969.99	24,24	16	4309	2632.47	24,24	14	4124	2479.77
C3540	12,12	20	5248	3500.17	12,12	19	5029	3336.12	12,12	18	4901	3204.69
C1908	8,8	16	1898	3052.14	9,9	13	1780	2481.34	9,9	12	1721	2362.53
C6288	13,13	15	4220	2647.50	13,13	12	3402	2225.80	13,13	11	3257	2063.59
C7552	20,20	19	11363	3139.88	20,20	17	9576	2809.44	20,20	15	9446	2519.44
alu2	10,10	15	2597	2758.33	10,10	13	2297	2454.37	10,10	12	2291	2318.60
alu4	13,13	16	4783	2809.70	13,13	15	4436	2647.50	13,13	14	4275	2502.75
des	32,32	21	31986	3297.93	32,32	20	31787	2652.17	32,32	19	30413	2509.63
Average	14,14	16.8	6329.4	3044.5	14,14	14.7	5948.5	2650.6	14,14	13.6	5735.2	2505.9

Table 2: Cluster size 8

Circuit	T-VPack+VPR				VPR+depopulation				*i*RAP			
	nx,ny	Channel	WL	Trans.	nx,ny	Channel	WL	Trans.	nx,ny	Channel	WL	Trans.
C432	4,4	18	579	6837.67	4,4	14	520	5474.81	4,4	13	489	5164.53
C499	5,5	20	758	6812.52	5,5	15	706	5276.55	5,5	13	693	4726.65
C880	5,5	20	1062	6812.52	5,5	18	950	6267.02	5,5	17	911	5984.84
C2670	12,12	44	4787	10681.2	12,12	42	4417	10252.5	12,12	41	4346	10030.7
C3540	9,9	33	5120	8795.07	9,9	25	4119	6833.04	9,9	24	4036	6611.48
C1908	7,7	24	1735	7061.06	7,7	15	1593	4712.85	7,7	15	1507	4712.85
C6288	9,9	21	3436	5922.69	9,9	17	2840	5012.35	9,9	16	2820	4649.46
C7552	12,12	41	10250	10030.7	12,12	39	8641	9601.38	12,12	38	8566	9393.22
alu2	7,7	18	1681	5598.02	7,7	16	1599	4962.91	7,7	15	1534	4712.85
alu4	9,9	24	3881	6611.48	9,9	23	3678	6377.87	9,9	21	3515	5922.69
des	21,21	53	29280	11660.3	21,21	49	27515	10880.6	21,21	47	27216	10470.0
Average	10,10	28.7	5688.1	7893.1	10,10	24.8	5143.5	6877.4	10,10	23.6	5057.5	6577.2

Table 1 shows the results for cluster size 4. Column 1 lists the 11 MCNC benchmark circuits. The next 4 columns show the array size, the minimum channel width, total wire-length and routing transistor count/per block when these circuits are placed and routed using VPR[3][17]. No clustering and placement modifications were done in this case. The next 4 columns show the results for the same circuits when we use our Rent's parameter-based clustering technique on top of the VPR tool. We achieve on average a 13% reduction in tracks per channel over designs clustered and routed without modifications. The last four columns show the same circuits after they have been clustered, placed and routed using the techniques outlined in sections 4 and 5. Over stand-alone VPR (first 4 columns), an average track reduction of 21.3% is achieved. Total wire-length is also reduced by 9.4%. Table 2 shows the results for size 8 clusters. In this case, for the same array size for each placed and routed circuit, we are able to reduce the average channel width by 17.7% over VPR. This translates into average savings of 11% in total wire-length over VPR. These results show the efficiency of our techniques and validate our claim that significant area savings can be accomplished by going for less than 100% logic utilization and efficiently using an inter-connect aware clustering and placement technique. We also observe that despite a slight increase in the number of clusters generated using our clustering technique, this increase does not translate into an increase in the critical path delay after routing. This can be attributed to a better distribution of the blocks during the placement process. Currently, we are exploring cost functions that could lead to better clustering results.

7. Conclusions and Future work

We have presented efficient routability-driven clustering and placement techniques for hierarchical FPGAs using Rent's rule and show that these techniques can result in significant area savings. By clustering circuits using Rent's parameter and minimizing the sum of differences between the architecture and design Rent's parameter, we have reduced the average track count for 11 MCNC benchmark circuits by 21% over a state-of-art FPGA place and route tool. Since as much as 90% of an FPGA is devoted to interconnects, this results in significant overall device area savings. Currently, we are considering the effects that different segment type and architectures will have on our results.

8. References

[1] M.J. Alexander, J.P. Cohoon, J.L. Ganley, G. Robins,"An Architecture-driven approach to FPGA routing using multi-weighted graphs", Proc. Euro-DAC, 1994, pp.259-264.

[2] E. Bozogzadeh, S. Ogrenci-Memik, M. Sarrafzadeh, "RPack: Routability-driven Packing for Cluster-Based FPGAs", Proceedings, Asia-South Pacific Design Automation Conference, January 2001.

[3] V. Betz, J. Rose "VPR: A New Packing, Placement and Routing tool for FPGA research", Proc Seventh FPLA, pp. 213-222, 1997.

[4] V. Betz, J. Rose, "Cluster-Based Logic Blocks for FPGAs:Area-Efficiency vs. Input sharing and Size", IEEE Custom Integrated Circuits Conference, 1997,pp.551-554.

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

[6] S. Brown, J. Rose, Z.G. Vranesic, "A Stochastic model to predict the routability of Field programmable gate arrays", IEEE Trans. on CAD, Dec. 1993, pp.1827-1838.

[7] P.K. Chan, M.D. Schlag, J.Y. Zien, "On routability prediction for field programmable gate-arrays,Proc. Design Automation Conf. 1993, pp.326-330.

[8] J. Cong, and Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs". IEEE Trans. on Computer-Aided Design, Jan. 1994, Vol. 13, No. 1, pp. 1-12.

[9] A. DeHon "Balancing Interconnect and Computation in a Reconfigurable Array (or why you don't really want 100% LUT utilization", Proc. FPGA 1999.

[10] S. Devadas, "Optimal Layout via Boolean Satisfiability", Proc. ICCAD, 1989, pp.294-297.

[11] W.E. Donath, "Placement and Average Interconnect requirements of Computer logic", IEEE Trans. Circuits and Systems, CAS-26:272-277, 1979.

[12] A.A.El Gamal, "Two-dimensional stochastic model for interconnections in master-slice integrated circuits", IEEE Trans. on Circuits and Systems,CAS_28:127-138,Feb, 1981

[13] L. Hagen, A.B. Kahng, F.J. Kurdahi, "On the Intrinsic Rent Parameter and Spectra-based Partitioning Methodologies", IEEE Trans. on CAD, pp.27-36, Vol. 13,No. 1, Jan 1994.

[14] S. Kirpatrick, C.D. Gelatt, M.P. Vecchi, "Optimization by Simulated Annealing", Science, 220:671-680, 1983.

[15] B.S. Landman, R.L. Russo, "On a pin versus block relationship for partitions of logic graphs", IEEE Trans. on Computers, C-20:1469-1479, 974.

[16] H. Van Marck, D. Stroobandt, J. Van Campenhout, "Towards an extension of Rent's rule for describing local variations in interconnect complexity" Proc. 4th Intl. Symposium for Young Computer Scientists, pp.136-141, 1995.

[17] A. Marquardt, V. Betz and J. Rose, ``Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density," ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, CA, February 1999, pp. 37 - 46.

[18] D. Stroobandt, P. Verplaeste, J. Van Campenhout, "Generating synthetic benchmark circuits for evaluating CAD tools", IEEE Trans. on CAD, 19(9):1011-1022, September 2000.

[19] R.G. Wood, R.A. Rutenbar, "FPGA routing and routability estimation using Boolean Satisfiability", Proc. Intl. Symposium on FPGAs" Feb 1997.

[20] Y.L. Wu, S.Tsukiyama, M. Marek-Sadowska, "Graph based analysis of 2-D FPGA routing" IEEE Trans. CAD, (1):33-44, Jan 1996.

[21] http://www.xilinx.com

[22] http://www.altera.com