Efficient Circuit Clustering for Area and Power Reduction in FPGAs

Amit Singh, Malgorzata Marek-Sadowska
Department of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, CA 93106
asingh@cornet.ece.ucsb.edu mms@ece.ucsb.edu

Abstract

We present a routability-driven bottom-up clustering technique for area and power reduction in clustered FPGAs. This technique uses a cell connectivity metric to identify seeds for efficient clustering. Effective seed selection, coupled with an interconnect-resource aware clustering and placement, can have a favorable impact on circuit routability. It leads to better device utilization, savings in area, and reduction in power consumption. Routing area reduction of 35% is achieved over previously published results. Power dissipation simulations using a buffered pass-transistor-based FPGA interconnect model are presented. They show that our clustering technique can reduce the overall device power dissipation by an average of 13%.

1. Introduction

Advances in Deep Sub-Micron (DSM) technologies have made possible million-gate FPGA devices from leading commercial vendors. Such high logic density, however, comes at an enormous interconnect cost since most of the device area (70-80%) is 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. Since most FPGAs are hierarchical in nature, circuit clustering (from henceforth, we use the terms clustering and packing interchangeably) has become an integral part of the FPGA synthesis flow. Current published works on FPGA clustering mainly focus on achieving tightly packed clusters for maximum logic utilization. While recent works [1][18] have attempted to address FPGA routability issues during the clustering phase, they have not fully addressed issues of circuit connectivity and underlying circuit structures, leading to inferior results.

Area, power, and performance remain the three most important factors that have restricted mass acceptance of FPGAs. We address area and power issues in this paper. Timing issues can also be adequately addressed by our technique, but are not the focus of this paper. We show that careful interconnect planning which matches the design and device complexities *apriori*, can lead to substantially less stress on routing, and can favorably impact both area and power. We present a connectivity-based bottom-up clustering technique which packs closely connected components together, achieves spatial uniformity in the clustered design using *Rent's Rule* [10][16], and reduces the external routing requirement in clustered FPGAs. This can positively impact FPGA compile times without the need for multiple iterations.

FPGA'02, February 24-26, 2002, Monterey, California, USA

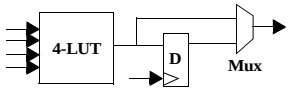
The paper is organized as follows. In Section 2, we present an overview of different bottom-up clustering techniques used both in the ASIC and FPGA design flows. The clustered FPGA model and the problem statement are presented in Section 3. The next section explains our connectivity-based bottom-up clustering technique. There, we also summarize new developments in interconnect planning in clustered FPGAs using Rent's rule and show how clustering and placement can be guided by this rule. Experimental results are presented in Section 5. Impact of power dissipation is discussed in Section 6. This is followed by conclusions in Section 7.

2. Previous Work

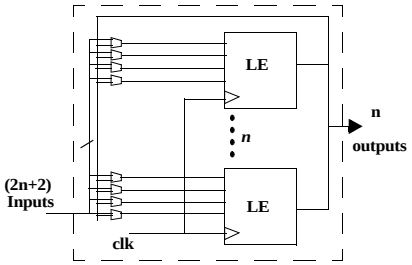
Clustering has traditionally been used in the VLSI industry to extract underlying circuit structures and construct a natural hierarchy in the circuits. A cluster can be viewed as a group of strongly interconnected nodes in a circuit such that the number of edges connecting these nodes to one another is much greater relative to the number of edges connecting this subset of nodes to the remaining nodes in the circuit. Clustering heuristics are classified as either bottom-up [5][7][8][11][13] or top-down [12][13]. Top-down approaches partition a given netlist into smaller subclusters. Bottom-up approaches start with a basic circuit cell as a seed and build clusters around the seed until certain area thresholds are met. Several bottom-up approaches have been used for standard cell designs, including the *cluster density* metric [13], *degree/separation* technique[5], *k-l* connectedness [11], and the *random walk* approaches [5]. While bottom-up approaches typically suffer from limitations because they are based on local connectivity information, they are faster than the top-down approaches. Additionally, the area and pin threshold requirements in FPGAs can be easily met during bottom-up clustering.

FPGAs usually consist of small, configurable Lookup Table (LUT)-based Logic Elements (LEs) connected by rich programmable interconnects. Since routing resources grow faster than on-chip logic resources, routing resources account for the major portion of the device's overall area and delay. Speed and area-efficiency of an FPGA are directly related to the granularity of its logic block. While coarse-grained blocks are very area-inefficient and have long internal logic delays, they can reduce the placement and routing stress by having fast local routing and significantly reduced external routing. Recently, FPGA vendors have introduced hierarchical FPGAs consisting of logic clusters. Examples of such devices are the Xilinx Virtex [22] and the Apex [23] from Altera. In these architectures, groups of Logic Elements (LEs) are clus-

tered to provide better performance. Figure 1 shows a typical FPGA Logic cluster composed of several LEs. Each LE is a 4-input LUT. Dedicated routing is provided inside each cluster for communication between the local LEs.



(a) Logic Element



(b) A Logic cluster of size *n*

Figure 1: Logic Cluster

Prior research [3][4] on clustered FPGA architectures has focussed mainly on the area-delay trade-off stemming from the size and structure of the clusters. Betz et al. [3] proposed a packing/clustering algorithm, *VPack*, for hierarchical FPGAs. The main idea in their work was to pack a technology-mapped circuit into clusters of a given size and input/output pin constraints. Their objective was to minimize the number of clusters and the number of inputs per cluster. The same authors introduced a tool, *T-VPack* [18], utilizing a timing-driven packing approach based on the idea of packing blocks on timing-critical paths to exploit fast local interconnects. The clusters generated using T-VPack use an average of 12% fewer tracks than the clusters generated using *VPack* for the same array size. A recent work, *RPack*, [1] presented a routability-driven packing algorithm which first identifies routability factors, prioritizes these factors into an improved clustering cost function, and achieves fewer routing tracks than *VPack*. This approach, however, produces routing track counts comparable to those generated by *T-VPack*. More recently, an interconnect-resource-aware clustering and placement technique was presented in [20] which used the clustering platform in T-VPack and achieved improved routability by clustering and placement with additional interconnect-resource constraints.

In this paper, we present a connectivity-based clustering technique that uses a degree/separation based connectivity analysis to build clusters for FPGAs. We show that LE seed selection during the clustering phase, and accurately modeling the clustering cost function, can greatly impact the final routability result and improve it by as much as 62% over the previously reported results in RPack [1]). We analyze the savings in power consumption and show that savings of as much as 13% can be achieved as a direct result from our clustering technique.

3. Preliminaries

In this section, we describe our architecture model and define some key terms. This is followed by a formal problem definition.

3.1 Architecture model

Figure 2 shows the features and a canonical Logic Cluster (LC) tile of a typical clustered FPGA device. LCs are groups of *n* LEs as shown in Figure 1. Local routing is provided inside the LCs, 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 assumed to be buffered 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 [4]. 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. All LCs are arranged into a two-dimensional array. Segmented wires are arranged on tracks, and programmable switches provide the connectivity between LCs.

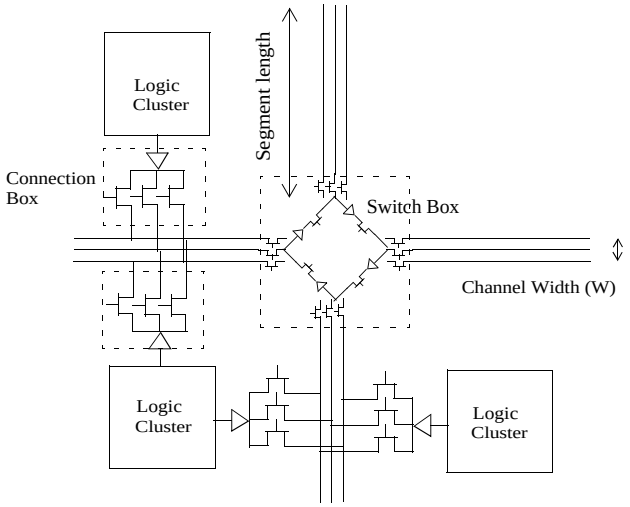


Figure 2: Architecture and Routing model

3.2 Key Terms

An input netlist is represented as a directed weighted graph $G(V, H, w)$, with vertex set V (corresponding to LEs), hyperedge set H, and a positive hyperedge weight $w(e)$ assigned for each $e \in H$. Each hyperedge connects all the vertices on an r pin net. Hyperedge weights are assigned to be $w(e) = \frac{2}{r}$, where r is the number of terminals on the net. Consequently, higher weights are assigned to smaller nets. We define the *degree* of an LE as the number of nets incident to that LE. The *separation* of an LE is the sum of all terminals of nets incident to the LE. A *connectivity factor (c)* is associated with each LE and is defined by the ratio:

$$c = \frac{separation}{degree^2} \qquad (1)$$

Smaller *c* values signify that more LEs are located in a given LE's neighborhood.

We characterize the complexity of a cluster through the well-know exponential relationship in *Rent's* rule [10][16]:

$$N_{io} = KB^{p} \qquad (2)$$

where N_{io} = number of pins in a cluster, B is the number of LEs in the cluster, K is the average number of connections per LE in the cluster, and p ($0 < p < 1$) is a parameter known as *Rent's parameter(exponent)*. Rent's rule is an empirical metric used to quantify circuit complexity. In the 1960s, several researchers independently found that the log-log plots of number of pins versus number of clusters in a logic design tend to form straight lines and follow the relationship in (2). Smaller values of p mean that the cluster's external routing requirement is low. A good clustering solution, therefore, will ensure that the Rent's parameter of the generated clusters is small.

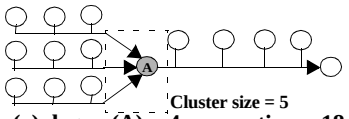
Network utilization is defined as the ratio of used interconnect to the available interconnect in an FPGA fabric in which the minimum number of tracks is the required channel width (W). We determine the FPGA architecture's Rent's exponent P_a from the log-log relationship between the number of available pins on a cluster versus the cluster size. It can also be calculated using formula (2). P_a captures the interconnect resource growth of an FPGA [9]. Since our architecture has uniform interconnect resources, P_a at the local level is assumed to be uniform. To check the accuracy of the P_a value calculated using formula (1), we generated uniform complexity benchmarks circuits [21] having values of P_d ranging from 0.2 to 0.9. Each generated benchmark was clustered, placed and routed, and the routing resource utilization was measured. We obtain P_a by drawing the best-fit line and using the value that gives the highest network utilization for the same architecture. The empirically calculated value of P_a is a good approximation of the one calculated using formula (2).

We now give a formal problem definition:

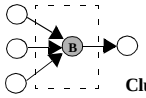
Problem Statement: *Given a 2-D mesh hierarchical FPGA chip with Rent's exponent P_a, and a design mapped to k-LUTs, cluster the mapped design so that the area and power of the clustered, placed and routed design are minimized, subject to the constraint of a given bounding box aspect-ratio.*

4. Clustering

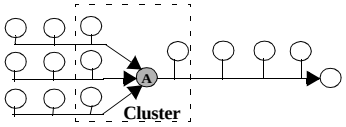
In this section, we present the routability-driven bottom-up clustering technique. Our technique aims to alleviate routing congestion for clustered FPGAs by (*i*) absorbing as many small nets into clusters as possible, and (*ii*) depopulating clusters according to Rent's rule in order to achieve spatial uniformity in the clustered netlist [20]. The clustering objectives are to minimize the number of external nets that need to be routed, and to match the Rent's parameter of the clustered design to that of the underlying FPGA architecture. Clustering is done in 2 phases. In the first phase, we find the *c* factor of all unclustered LEs according to equation (1). We maintain an array of lists for storing all unclustered LEs. Each element in the array is a list containing all unclustered LEs with the same degree in ascending order of their *c* values. These sorted lists speed up the search process for unclustered LEs during the second clustering phase.



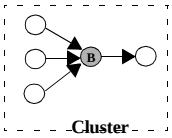
(a) degree(A) = 4, separation = 18, *c* = *1.125*



(b) degree(B) = 4, separation = 8, *c* = *0.5*



(c) Number of nets absorbed = 0



(d)Number of nets absorbed = 4

Figure 3: Cluster seed selection -- B is a better seed LE

The second phase sequentially builds clusters. An LE seed with the highest *degree* and lowest *c* value will be selected to be the seed of a new cluster. The advantage of our degree/separation metric in choosing a seed is illustrated in Figure 3. Let *A* and *B* be two candidate seed LEs. Both *A* and *B* have the same *degree*, i.e. *degree*(*A*) = *degree*(*B*) = 4. Here, LE *B* is a better seed candidate since its *c* value is smaller than that of *A*. In other words, *B* has a lesser average separation from its neighboring nodes, and more nets can be potentially absorbed for internal routing. This is evident from figure 3.d, in which all 4 nets incident to *B* have been absorbed. Choosing *A* as a seed LE results in no nets being fully absorbed inside the cluster. Choosing cluster seeds in this manner consistently produces better routable circuits as measured by the number of tracks needed to route the circuit.

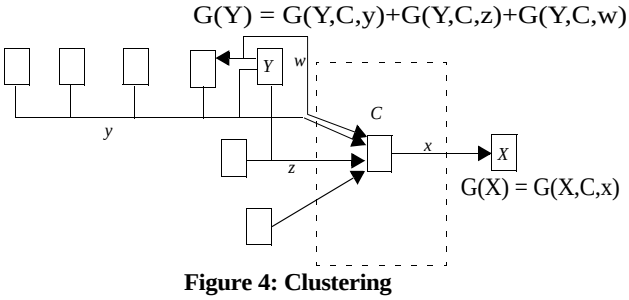


Figure 4: Clustering

Once a cluster seed is chosen, unclustered LEs connected to this seed LE are assigned gain values according to their attraction to the cluster. The unclustered LE with the highest gain is absorbed into the cluster. Consider an unclustered candidate LE *X* which is being considered for inclusion into a currently open cluster *C* as

shown in Figure 4. The attraction of X to C is a function of the weight of the net x and the number of pins of x already inside cluster C. We define this gain quantitatively as:

$$G(X, C, x) = 2nw(x) \times (1 + \alpha_x) \qquad (3)$$

where α_x = number of pins of net x already inside C, n is the cluster size, and $w(x)$ is the weight of net x ($w(x) = 2/r$ where r is the number of pins on x). If X and C have more than 1 net in common (LE Y in figure 4), then the gain of X is the sum of gains contributed by these common nets. If adding X to C fully absorbs net x (as is the case in figure 4), then $G(X, C, x)$ is multiplied by a constant k: $G(X, C, x) = k[2nw(x) \times (1 + \alpha_x)]$, where k>10. This essentially ensures that blocks attached to the currently open cluster through smaller nets are more likely to be absorbed in the cluster since they are assigned higher weights. Assigning higher weights to such LEs is essential, as there can be other unclustered LEs attached to C (for example, LE Y) through multiple long nets which may not be entirely absorbed in C. In figure 4, we need to ensure that $G(X) > G(Y)$.

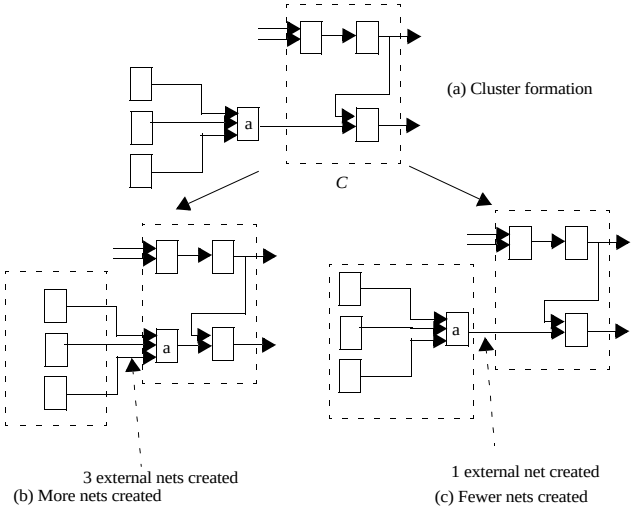


Figure 5: Reducing external routing demand during clustering

An unclustered block can be absorbed into an open cluster only if cluster size and pin constraints are satisfied. In order to guarantee spatial uniformity of the clustered netlist, we limit the number of available pins using Rent's rule. Our aim is to guarantee that the Rent's parameter of any cluster is no more than the Rent's parameter of the underlying architecture, P_a. The available 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.

$$j = (k + 1)n^{Pa}$$

$$k + 1 \le j < 3n + 2 \qquad (4)$$

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 [20]. The spatial uniformity requirement can potentially lead to unused cluster pins and less than 100% cluster utilization. However, as figure 5 shows, this in fact can reduce the number of external nets. Reducing the number of external nets greatly impacts the final routability results. In figure 5, cluster C can absorb an additional LE. LE a has the highest gain and adding a to C does not violate either the architecture pin or cluster size constraint. However, adding a to C creates 3 external nets and increases the cluster's Rent's parameter. Our interconnect-resource-aware clustering constraint recognizes this fact and makes sure that the situation shown in figure 5.b does not arise. Instead, LE a is chosen as the seed for a new cluster, thereby creating only a single external net as shown in figure 5.c. We end up with 2 clusters in both figures 5.b and 5.c -- yet the clustered netlist in figure 5.c is more routable. The preceding example shows that instead of striving to achieve more than 98% [4] cluster logic utilization, limiting the number of usable cluster pins can potentially lead to uniformity in the clusters generated. The intent is to keep a low ratio of used pins to total number of pins in the logic cluster. Better still would be to consider this aspect during the architecture design phase itself.

Table 1: External nets : Cluster size 8

Circuit	T-VPack[18]		RPack[1]		*Ours*	
	Clusters	Ext. nets	Clusters	Ext. nets	Clusters	Ext. nets
alu4	192	804	196	985	196	624
apex2	240	1249	242	1288	249	993
apex4	165	863	167	868	168	739
bigkey	214	1040	214	1060	227	585
clma	1054	5307	1054	5585	1089	3884
des	200	1214	202	1339	352	1213
diffeq	189	1033	189	895	195	662
dsip	172	762	188	1219	228	472
elliptic	454	2247	462	2300	475	1408
ex1010	599	3110	602	3064	613	2575
ex5p	139	767	139	754	140	664
frisc	446	2048	447	1983	477	1521
misex3	178	840	178	876	191	679
pdc	582	2627	590	3011	608	2246
s298	243	767	243	1330	251	591
s38417	802	4423	802	3921	825	3153
s38584	806	4183	806	3556	839	2884
seq	221	1055	223	1166	223	878
spla	469	2099	473	2336	484	1771
tseng	133	801	133	764	141	535
Average	374.9	1861.9	377.5	1915.0	398.5	1403.8
change (%)	0.95	1.33	0.96	1.36	1.0	1.0

Table 1 shows the number of clusters and the number of external nets created using our clustering technique. We also show results from previously published FPGA clustering techniques [1][18]. Both T-Vpack and RPack produce 30% more external nets when compared to our technique. In fact, our clustering technique is able to absorb more low-fanout nets inside clusters. Section 5 shows how this translates into significant area savings in terms of the number of routing tracks used.

Once clustering has been done, the original netlist has been reduced to a new netlist with each node corresponding to a cluster. We then use an interconnect-aware placement tool that we have developed [20] to place this hierarchical netlist. Note that during placement, no cluster modification is made. A fairly uniform local *complexity* of the clusters does not guarantee uniformity of inter-cluster interconnect length distributions during placement. The need for spatial uniformity in the placed design necessitates the incorporation of Rent's parameter in the placement tool. This spa-

tial uniformity is desired between the clusters themselves and not inside the clusters where placement and routing are assumed. Recently, we proposed a placement tool which uses a local Rent's parameter weighted summation of net-lengths as a component of the cost function [20]. The placement cost function used is:

$$\sum_{i=1}^{Nets} q(i)\left(1 + \sum_{\forall(k \in i)} |P_{dk} - Pa|\right)(wirelength_i) \qquad (5)$$

P_{dk} corresponds to the *local placed* Rent's parameter of the net *i* and the clusters connected to it. 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

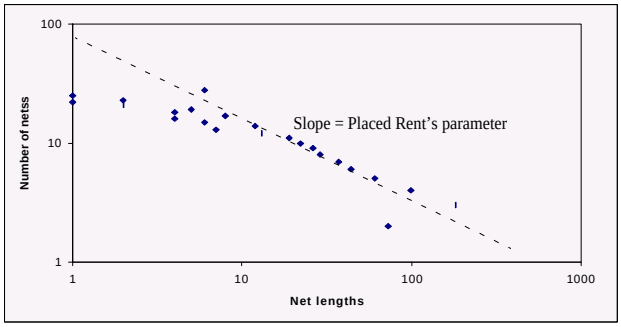


Figure 6: Circuit *dsip*: Placed Rent's parameter

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. Figure 6 shows a log-log plot of the number of nets versus the net lengths of a placed circuit. The Rent's parameter of this placed design is calculated as the slope of the best fit line [17].

5. Experimental Results

We have implemented our clustering technique, named *i*RAC (*i*nterconnect Resource Aware Clustering) on top of VPRv4.30 [2] along with our interconnect-aware placement algorithm, *i*RAP [20]. We placed and routed 20 of the largest MCNC random benchmark circuits on clustered FPGAs. Cluster size 8 was used in our experiments. The number of I/O pads per row/column [4] in the array is found by:

$$Pads = \lceil 2\sqrt{n} \rceil \qquad (6)$$

where *n* is the cluster size. Each circuit was mapped into 4-input LUTs using flowmap [8], clustered using the technique outlined in section 4, and placed using our Rent's rule aware placement technique [20].

Table 2 shows the routing track results for T-VPack [18], RPack [1], and our clustering and placement technique (both with and without using the degree/separation based seed selection), for the same array size, in increasing order of Rent's parameters of the clustered designs. Both T-VPack and RPack use about 35% more tracks than our combined clustering and placement routing techniques. More than 90% of the improvement is due to our effec-

Table 2: Routing tracks: Cluster size 8

Circuit	Array size	T-VPack[18]	RPack[1]	Random Seed	*i*RAC+*i*RAP		
	nx,ny	Channel	Channel	Channel	Channel	Pins used	*p*
bigkey	18,18	17	15	9	9	11.98	0.476
dsip	18,18	14	24	9	9	11.98	0.490
s38584	29,29	32	26	20	19	12.20	0.524
tseng	12,12	21	21	14	12	13.21	0.540
elliptic	22,22	37	32	28	24	13.97	0.561
s38417	29,29	29	25	20	19	13.66	0.562
diffeq	14,14	20	19	16	14	14.41	0.569
des	21,21	17	18	15	13	11.31	0.584
s298	16,16	20	29	18	15	15.52	0.596
Avg.(p < 0.6)		23	23.2	16.5	14.9	13.13	0.545
% change		1.55	1.56	1.11	1.0		
frisc	22,22	39	34	31	31	15.65	0.613
alu4	14,14	26	34	25	22	17.06	0.645
clma	33,33	47	48	42	36	17.39	0.649
misex3	14,14	29	32	26	24	17.40	0.674
spla	22,22	42	48	40	37	18.75	0.678
pdc	25,25	52	56	48	45	19.03	0.687
seq	15,15	33	37	33	30	18.93	0.695
ex5p	12,12	37	36	33	31	19.60	0.704
apex2	16,16	34	35	30	29	19.44	0.717
ex1010	25,25	41	42	33	29	19.43	0.726
apex4	13,13	37	35	33	31	19.66	0.727
Avg.(p > 0.6)		37.9	39.7	34	31.4	18.4	0.683
% change		1.20	1.26	1.09	1.0		
Overall Average	21,21	31.2	32.3	26.15	23.9	16.03	0.620
Overall %change	1.0	1.34	1.35	1.10	1.0	-	-

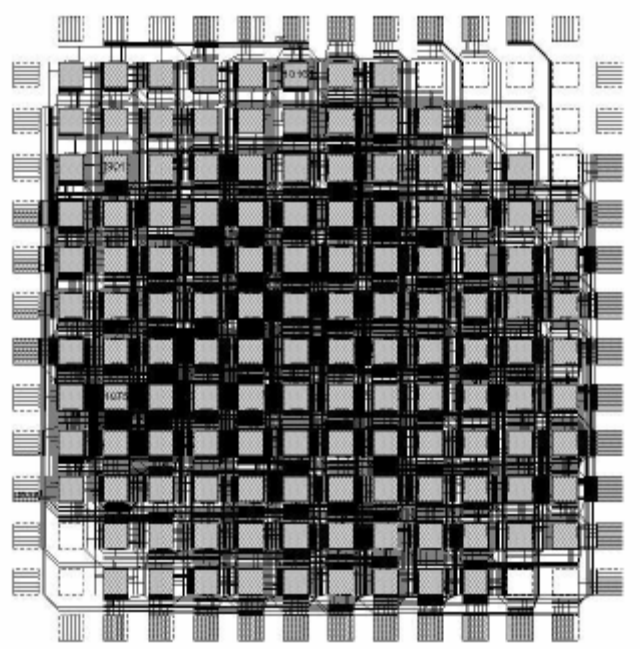
tive clustering technique. Since 70-80% of the total routing area is devoted to interconnects, this can result in overall die reduction of as much as 28%. For circuits with $p < 0.6$, T-VPack and RPack use an average of 55% more tracks than *i*RAC. The impact in terms of extra tracks used by T-VPack and RPack for circuits with $p < 0.6$ is less significant. The 5th column shows the routing track count for circuits clustered from random seeds chosen in a manner similar to that for RPack but with our modified gain cost function.This column shows that choosing seeds without considering the underlying structure can result, on average, in 10% more tracks to route them when compared to *i*RAC (column 6). The last 2 columns show the average number of pins used per cluster and the *Rent's* parameter of the clustered designs. *i*RAC is more effective in clustering circuits which have a higher percentage of low-fanout nets. By absorbing most of the low-fanout nets, *i*RAC is able to lower the number of external nets, and the Rent's parameter of the circuits after clustering. For example, table 3 shows the net profile for circuit *dsip* (p = 0.49), both before and after clustering. The first 2 columns show that most of the nets in the flat netlist are low-fanout nets. After clustering, the majority of the 2-pin nets are absorbed inside the clusters for internal routing. On the other hand, circuits which have fewer low-fanout nets end up with higher Rent's parameters after clustering. Note that our clustering technique never uses more than 20 pins on any of the clustered designs. Figure 7 shows circuit *tseng*, clustered, placed and routed

Table 3: Circuit *dsip*: Netlist profile

Before clustering		After clustering	
# Pins	*# Nets*	*# Pins*	*# Nets*
2	1141	2	16
3	221	3	221
4	1	4	3
5	39	5	38
6	184	6	188
7	4	224	1
9	2	225	1
10	1	226	2
225	2	228	2
450	2		
906	1		
908	1		

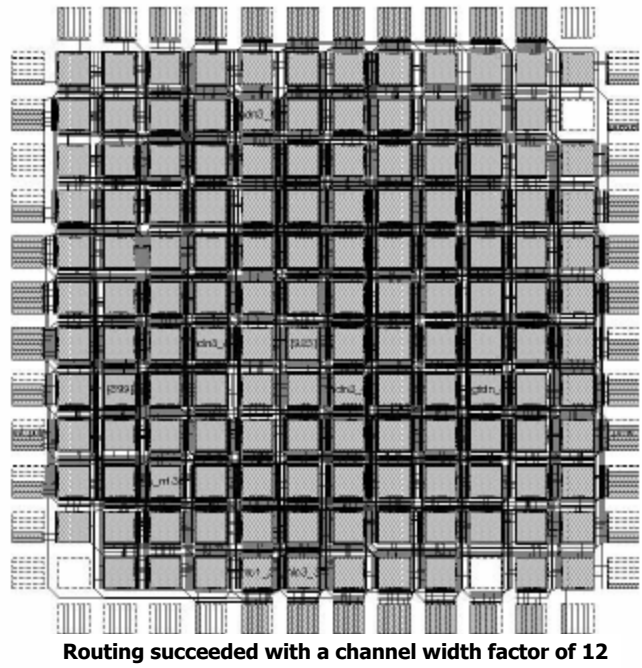
Table 4: Wirelength and Transistor count

Circuit	T-VPack		RPack		*i*RAC+*i*RAP	
	Wire-length	Trans.	Wire-length	Trans.	Wire-length	Trans.
alu4	7827	9.196e6	9830	1.199e7	6920	7.81e6
apex2	12888	1.559e7	13431	1.604e7	11846	1.331e7
apex4	9522	1.126e7	9214	1.066e7	8496	9.749e6
bigkey	7852	9.964e6	7473	8.79e6	3867	5.380e6
clma	65298	8.974e7	65997	9.162e7	53125	6.899e7
des	10500	1.351e7	10510	1.427e7	8675	1.037e7
diffeq	6105	7.118e6	5590	6.771e6	4478	5.221e6
dsip	6105	8.222e6	9841	1.394e7	3638	5.38e6
elliptic	22945	3.175e7	20401	2.747e7	16724	2.071e7
ex1010	34306	4.524e7	35352	4.632e7	27282	3.211e7
ex5p	7877	9.632e6	7885	9.376e6	7171	8.078e6
frisc	24584	3.344e7	22300	2.921e7	21061	2.662e7
misex3	8684	1.024e7	8873	1.127e7	7776	8.849e6
pdc	44987	5.72e7	46757	6.155e7	39149	4.958e7
s298	7554	9.253e6	10152	1.331e7	6497	7.42e6
s38417	30675	4.309e7	27594	3.725e7	21250	2.849e7
s38584	33831	4.746e7	27104	3.871e7	22283	2.849e7
seq	11532	1.333e7	12207	1.492e7	10327	1.211e7
spla	28914	3.597e7	32068	4.104e7	26128	3.175e7
tseng	4824	5.517e6	4507	5.517e6	2826	3.193e6
Avg.	19340.5	2.535e7	19354.3	2.55e7	15475.9	1.92e7
change (%)	1.25	1.34	1.25	1.35	1.0	1.0



Routing succeeded with a channel width factor of 21

(a) RPack: Circuit *tseng*



Routing succeeded with a channel width factor of 12

(b) Ours: Circuit *tseng*

Figure 7: Circuit *tseng:* Clustered, placed and routed

using RPack [1] (we use RPack since it's routing result is the same as that of T-VPack in this circuit) and our approach. In both the cases, we use a routability-driven placer and router. Each block in the figures is a size 8 cluster with each cluster containing 18 input and 8 output pins. As can be seen from the figure, our clustering and placement solution are able to reduce the overall track count by over 42% for the same array size.

Table 4 shows the total wire-lengths and total transistor count for T-VPack [18], RPack [1] and our connectivity-driven clustering and placement technique. Our clustering and placement technique is able to reduce the average wire-length by 25%, and as will be evident in the next section, this reduction is directly related to the amount of interconnect power savings associated with our technique. Furthermore, we can effectively reduce critical delay in a majority of the benchmarks by using our Rent's rule aware placement technique. More work in performance-driven clustering and placement is currently underway.

6. Power consumption

Dynamic power consumption is a major concern in FPGAs. FPGAs consume significantly more power than their ASIC counterparts. For example, a Xilinx 4000 series device can consume as much as 1000nW/MHz/gate whereas its ASIC counterpart consumes only 20-30nW/MHz/gate. Moreover, power consumption for commercial FPGAs grows linearly with an increase in frequency. For example, Xilinx[22] estimates that a Virtex device clocked at 250MHz and consisting of 13000CLB slices can consume more than 21W of power.

By reducing the overall routing area and wire-length, we can directly impact power dissipation. If P_{total} is the total power dissipated in a device, then the 4 major components of power dissipation are:

$$P_{total} = P_{interconnect} + P_{logic} + P_{clock} + P_{control} \quad (7)$$

Since we do not increase the array size, and we assume no clock gating, the last 2 components in (7) remain fairly constant. Hence, any change in power dissipation is mainly due to changes in $P_{interconnect}$ and P_{logic}. Therefore:

$$\Delta P_{total} = \Delta P_{interconnect} + \Delta P_{logic} \quad (8)$$

ΔP_{logic} can be assumed to increase slightly (~5%) from our clustering technique, since the average number of active LCs, as shown in Table 1, increases by around 5%. However, ΔP_{total} is mostly a function of the total wire-length of the routed circuit, i.e.

$$\Delta P_{total} \approx \Delta P_{active-segements} + \Delta P_{inactive-segements}$$

Table 5: Active power dissipation--0.18µ technology

Segment	Power (mW/segment)
Active(loaded)	0.54mW
Inactive(unloaded)	0.0385mW

Table 5 shows the total power contribution by both a loaded and unloaded wire-segment (Figure 2) in a 0.18µ 1.8V technology at 250MHz. We have lumped R and C values of wire-segments of length 1 and the associated pass transistors, buffers, parasitics, and capacitive loading in these simulations. This table clearly shows that active wire segments contribute the majority of the total interconnect power consumed. From table 3, our clustering and placement technique is able to reduce the average wire-length by around 25%. Assuming that interconnect and logic resources consume approximately 70% of the total dissipated power, and that interconnects occupy around 70-80% of the total device area, we can save approximately 13% of the total device power by using our clustering and placement techniques.

7. Conclusions

We have presented an interconnect-aware connectivity-based bottom-up clustering technique for significant reductions in area and power for hierarchical FPGAs. Circuits with smaller *Rent's* parameter (usually < 0.6) and fewer high-fanout nets show the most improvement from our technique. This can be attributed to smaller nets being absorbed inside the clusters for internal routing. For cluster size 8, area saving of as much as 62% (35% on average) is achieved over previously published results [1][18]. This saving directly impacts the overall device power consumption and results in approximately 13% power saving. We have observed that area and power savings can be more significant for clusters of bigger sizes. An effective seed selection during the clustering phase, coupled with a cost function that accurately models the attraction of unclustered LEs to a cluster, can have a major impact in reducing routing stress. Incorporating interconnect resource constraints, during both clustering and placement, can further reduce total routing demand. We are currently extending our integrated techniques to multi-level clustering and are also looking at performance improvements.

8. Acknowledgment

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

9. References

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

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

[3] 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.

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

[5] J. Cong, L. W. Hagen, and A.B. Kahng, "Random Walks for Circuit Clustering", Proc IEEE Conf. on ASIC pp. 14.2.1 - 14.2.4, June 1991.

[6] 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.

[7] J. Cong and M. Smith, “A Parallel Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design”, Proc Design Automation Conf, 1993, pp. 755-60.

[8] J. Cong and S. K. Lim, “Edge Separability based Circuit Clustering With Application to Circuit Partitioning”, IEEE/ACM Asia South Pacific Design Automation Conference, pp. 429-434, 2000.

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

[11] J. Garbers, H.J. Promel, and A. Steger, “Finding Clusters in VLSI Circuits”, ICCAD 1990, pp. 520-523.

[12] L. W. Hagen and A. B. Kahng, “Combining Problem Reduction and Adaptive Multi-Start: A New Technique for Superior Iterative Partitioning”, IEEE Transactions on Computer-Aided Design, 1997, pp. 709-717.

[13] D. J.H Huang and A. B. Kahng, “When Clusters Meet Partitions: New Density-Based Methods for Circuit Decomposition”, Proc European Design and Test Conf, 1995, pp. 60-64.

[14] A. B. Kahng and R. Sharma, “Studies of Clustering Objectives and Heuristics for Improved Standard-Cell Placement”, UCLA CS Dept report, January 1997.

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

[16] 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.

[17] 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.

[18] 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.

[19] G. Parthasarathy, M. Marek-Sadowska, A. Mukherjee, A. Singh, “Interconnect-Complexity Aware Placement for FPGAs using Rent’s Rule”, Proc. of 3rd System Level Interconnect Prediction Workshop, April 2001.

[20] A. Singh, G. Parthasarathy, M. Marek-Sadowska, “Interconnect-Resource Aware Placement for Hierarchical FPGAs”, Proceedings of International Conference on Computer-Aided Design, November 2001.

[21] 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.

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

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