

Routability-driven Packing: Metrics and Algorithms for Cluster-based FPGAs

E. Bozorgzadeh[†]

S. Ogrenci Memik[†]

X. Yang[‡]

M. Sarrafzadeh[†]

[†] Computer Science Department
University of California, Los Angeles (UCLA)
3531C Boelter Hall
Los Angeles, CA 90095, USA
e-mail: {elib,seda,majid}@cs.ucla.edu

[‡] Synplicity Inc.
600 W California Ave.
Sunnyvale, CA 94086
email: xjyang@synplicity.com

ABSTRACT

Most of an FPGA's area and delay are due to routing. Considering routability at earlier steps of the CAD flow would both yield better quality and faster design process. In this paper, we discuss the metrics that affect routability in packing logic into clusters. We are presenting a routability-driven clustering method for cluster-based FPGAs. Our method packs LUTs into logic clusters while incorporating routability metrics into a cost function. Based on our routability model, the routability in timing-driven packing algorithm is analyzed. We integrate our routability model into a timing-driven packing algorithm. Our method yields up to 50% improvement in terms of the minimum number of routing tracks compared to VPack (16.5% on average). The average routing area improvement is 27% over VPack and 12% over t-VPack.

Keywords: VLSI CAD, Field Programmable Gate Arrays (FPGAs), Technology mapping, Clustering Techniques, Optimization, Algorithm.

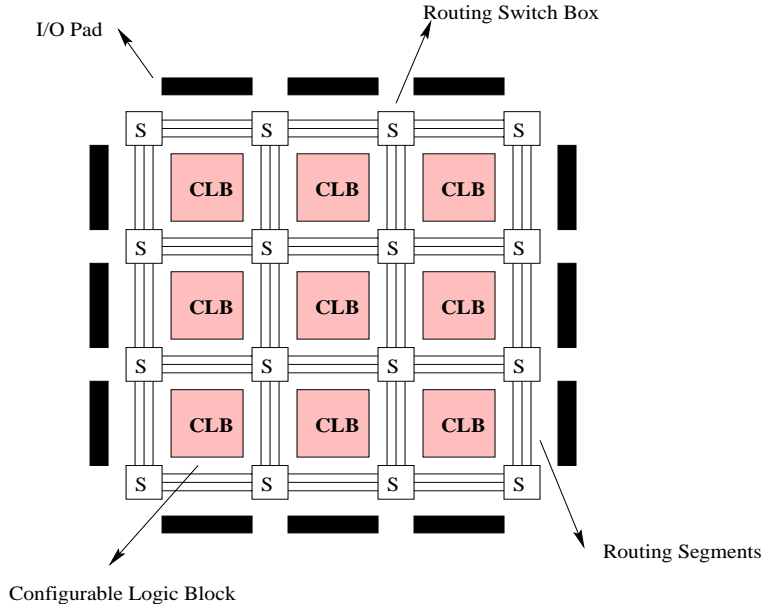


Figure 1: Island style FPGA

1 INTRODUCTION

Today's technology allows FPGAs to be designed as multi-million system gate devices at the heart of electronic systems. Since FPGA is an integral part of many digital systems, the significance of optimization problems in mapping circuits on FPGA has increased. There are two important issues related to the FPGA mapping process: the quality of the resulting mapping and the run-time of the tools serving in the process. The former being more dominant for FPGAs, both aspects are important. Similar to ASIC design, minimizing the delay is an important objective as well as minimizing the silicon area. Area of an FPGA consists of routing area and logic area. Optimizing the utilization of both routing and logic resources is very crucial to obtain a good quality result.

FPGAs consist of smaller configurable building blocks called logic blocks or Configurable Logic Blocks (CLBs), which are placed on the FPGA chip either on a two-dimensional array (see Figure 1) or in a set of rows. The CAD flow of mapping a circuit on FPGA consists of four major stages. In the first stage the circuit is basically logically optimized. In stage 2, the optimized circuit is divided into CLBs of the FPGA, which is called technology mapping. Placement and routing stages accomplish the assignment of subcircuits on CLBs and programming the routing switches of FPGA.

Due to highly constrained and discrete interconnect structure of current FPGAs, routing is a challenging problem. Most of the time current FPGA routers cannot use available routing resources efficiently. This leads to a large portion of the routing area to be wasted. Also, depending on the complexity of the particular design routing might require a fairly large amount of time, often several hours to be completed. Hence considering routability at earlier steps of the CAD flow would both yield a better quality of the result and

less design time in later stages.

FPGA vendors have different logic block configurations. There are two kinds of CLBs: LUT-based blocks and multiplexor-based blocks. LUT-based logic blocks are more popular. There have been several contributions in development and design of FPGAs towards reducing the gap in density and performance between ASIC and FPGA implementation. Hierarchical features have been added into logic and routing architecture of FPGAs. Many commercial FPGAs, such as Xilinx, Altera, and Actel FPGAs include logic blocks that contain several LUTs [1]. A collection of basic logic elements that are grouped together to be placed in one complex logic block is called a *cluster* (See Figure 2(a)). FPGAs with logic blocks containing multiple basic blocks are called *cluster-based* FPGAs. Each CLB (configurable logic block) is a cluster of basic logic elements in cluster-based FPGAs. The structure and granularity of the logic block have a significant impact on the area-efficiency and performance of the FPGA. If the logic block is fine-grained, the circuit to be implemented will be distributed over more number of logic blocks. This has a negative impact on routability, since more blocks need to be interconnected. Since the interconnect inside the logic blocks is hardwired, local interconnect can be made very fast and efficiently. This improves routability and decreases the load on the router significantly by reducing the size of problem. Two main benefits of clustering a basic block into CLBs are speed in compilation and circuit delay improvement. On the other hand, it is not feasible to increase the complexity of the logic blocks beyond a certain limit. If the logic blocks become too complex it becomes difficult to utilize them fully, hence several logic blocks will be wasted. Due to constraints on the number of input pins and the number of blocks within each cluster, all the resources in a cluster cannot be used in circuit implementation. The task of assigning basic logic blocks to clusters is called *packing*. Due to no accurate means to estimate the interconnect at logic synthesis level, it is not easy to deal with routability of circuit at logic level. However, if special properties of the interconnect available at logic level, such as sharing among the pins, can be exploited during packing logics into basic blocks, significant gains can be obtained in terms of routability. In the past routability at the packing stage has not been considered as extensively as it has been at the technology mapping stage. Packing can bring improvements on the routability, since after technology mapping a more accurate estimation on the interconnect is available.

In this paper we propose a routability-driven packing algorithm. We show improvements in routing area upon the state-of-the-art logic packing algorithms called VPack and t-VPack: Logic Block Packing Algorithm [4, 6]. We are introducing a new method to consider routability at the packing stage. Our method in selecting a block for clustering can easily be integrated with other clustering algorithms. We are demonstrating the effect of our method on the routability by synthesizing the benchmark circuits through the complete CAD flow. We have technology mapped a given circuit, then applied our routability-driven packing method for clustering, and finally placed and routed the circuit. We present the results of the final routing and show that our method improves the routability significantly. Our new algorithm, RPack, indeed improves routability compared to VPack. As our results on 20 largest MCNC benchmarks show in Section 5, we are able to improve the minimum required number of routing tracks by 16.5% on an average. A preliminary version of

this work appeared in [8]. We also integrated our routability function in timing-driven packing algorithm. Based on our routability model, routability in timing-driven packing algorithm is analyzed. Compared to t-VPack, the routing area is improved by 12% on an average.

The organization of the paper is as follows: Previous work on routability-driven technology mapping and algorithms for cluster packing are discussed in Section 2. Section 3 describes the FPGA architecture we are targeting, utilization and routability issues and problem formulation for the packing problem. In Section 4 RPack, our routability-driven packing method is described. Experimental results are presented in Section 5. Section 6 includes conclusions and future work.

2 Previous Work

Most commercial FPGAs use configurable blocks containing several LUT. Packing LUTs into clusters is an important design step introduced for cluster-based FPGAs. It can be viewed as a sub-task within technology mapping stage in which logic gates are assigned to LUTs and registers. We will first mention contributions made in the technology mapping area. The majority of research devoted to technology mapping has been done with the objective of improving either timing [7, 12, 18, 22, 13] or area-efficiency [19, 24, 20] or trade-off between depth and area [19]. Compared to the amount of the effort made in this area there is little work done in the routability driven technology mapping domain [15], [17]. The routability driven technology mapper for LUT-based arrays, Rmap [17], employs a mapping strategy that considers routability.

The packing problem is a clustering problem. Clustering has been studied extensively for various applications, such as placement [25], technology mapping [4, 17], etc. Packing is a clustering problem with constraints on the number of input pins and the number of LUTs in each CLB. The objective is to minimize the number of required CLBs to cover all the LUTs while satisfying the constraints. Betz and Rose proposed VPack and t-VPack, logic block packing algorithms [4] for cluster-based FPGAs. VPack and t-VPack are one of the best known packing tools for FPGAs. VPack first packs a flip flop and a LUT together into a basic logic element using a matching based method. Then these BLEs are packed in a greedy manner into logic clusters with the local optimization objectives being to fill each cluster to its capacity and minimize the number of used inputs to each cluster. This approach is inspired from [21]. In [6] a timing-driven packing tool for FPGAs, t-VPack is proposed. The blocks on the critical path are preferred to be packed together in a CLB so that the delay can be improved by exploiting local wiring in the CLB to route the critical nets. t-VPack delivers a better routability compared to VPack. Later, we will describe the routability potential in timing-driven packing algorithms. Also in [23], a packing approach is proposed based on maximum weight matching on circuit graph. Recently researchers in [9] have proposed a new technique for packing logic into clusters. Based on Rent's rule for each application, the connectivity of each cluster is defined. In this approach routability is weighted according to the connectivity of the application. It is a good idea to consider routability based on connectivity of the circuit. On the other hand, the weight of routability in the overall

optimization objective is fixed during clustering for each application. By this way routability cannot be considered accurately. In this work, we scale the weight of the routability factor dynamically. In [14], a good survey of packing methods for cluster-based FPGAs is presented.

In all these approaches, when a logic block is packed into an existing cluster, the type of nets being shared is not considered. An important issue in cluster-based FPGAs is the limited number of inputs. Therefore, considering the input/output pin sharing besides edge covering can improve the performance [8]. In this paper we analyze the issues during the packing process extensively. We are introducing new metrics that are used to form a new objective function to evaluate routability. We took the algorithm of VPack as a basis as we will describe in later sections and have built our own approach upon it.

3 PACKING IN CLUSTER-BASED FPGAS

In this section we will study the issues in packing stage of technology mapping for cluster-based FPGAs. Also the routability driven packing problem is formulated.

3.1 Cluster-based FPGA Architecture

The FPGA we are targeting is of the SRAM-based island-style structure. It contains a square matrix of logic blocks. Between each row and column routing tracks are located. The structure of the basic logic block is illustrated in Figure 2 (b). It contains a K -input LUT and one flip flop. A K -input LUT is able to implement any function of its K inputs (2^K functions). However size of look-up table grows exponentially with the number of inputs. It has been shown that LUT with input size 4 is the most area efficient configuration [14]. The logic cluster is shown in Figure 2 (a). The cluster size, N , is defined as the number of basic blocks contained in the cluster. The cluster takes I inputs that are connected to the LUTs inside basic blocks. Not all $4N$ basic block inputs are accessible externally. Only I out of these are connected to input multiplexors of the cluster. These input multiplexors allow any of the I inputs to be connected to any of the $4N$ basic block inputs. Also any output of N basic blocks can be connected to any basic block input through these multiplexors. The cluster contains N output pins connecting each basic block output to one cluster output. Similar structure is used in [14].

In packing stage of CAD flow for cluster-based FPGAs, the input circuit is represented in terms of LUTs and registers. As shown in Figure 2(c) if LUT l is followed by register r and there is no interconnection to any other elements from the net connecting LUT l and register r , they both can be implemented by a basic logic block shown in Figure 2(c). Otherwise each register or LUT should be assigned to one basic logic block. An optimal pattern matching-based method to pack the register-LUT pairs into basic blocks is proposed in [4]. Hence the problem is simplified to packing a set of basic blocks into clusters. We are focusing on clustering the basic blocks into logic clusters after each register and LUT are assigned to a basic logic block.

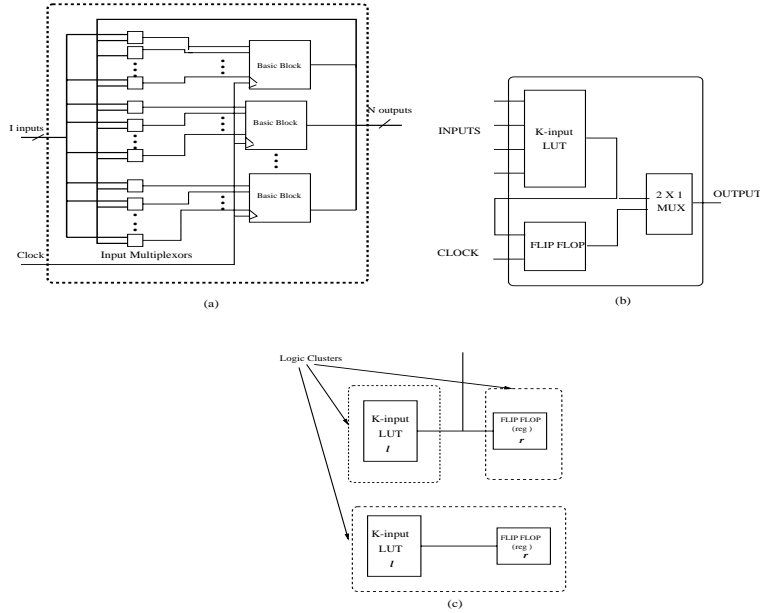


Figure 2: (a) Internal Structure of a Logic Cluster (b) Basic Logic Block (c) Matching LUT-register pairs [14].

3.2 Routability and Utilization

Routing is a very time consuming stage in CAD flow for both FPGA and ASIC Designs. A router can fail to route a logically optimized circuit. In addition, most of an FPGA's area and delay is due to routing. Since the routing tracks in each channel are fixed, routing in a congested channel is hard for FPGA router. *Routability* means how likely the circuit is routable at the end of CAD flow. Hence, considering routability in earlier stages can have significant impact in routing results. Incorporating routability in earlier stages in design flow does not necessarily guarantee the routability of a circuit. However it can be improved.

By clustering the logic blocks, the number of connections between clusters is reduced. In other words, the routability can be increased. Consequently, the routing area can be improved in terms of the number of tracks required in each routing channel and switch area.

There are several factors affecting routability at different CAD flow stages. In technology mapping and clustering stage the following factors can impact routability. The first three factors have been mentioned previously in [17]:

- Ratio of blocks used to total blocks available
- Number of pins per block
- Ratio of pins per net
- Ratio of used pins to total number of pins of the logic block
- Number of exposed nets

The first factor shows the impact of logic utilization on routability. If the blocks are not packed efficiently, a larger FPGA is required to implement a circuit and therefore a large routing area can be wasted. Utilization depends both on the architecture and packing method. The second factor depends on the architecture. The impact of different logic block architectures on FPGA performance has been studied earlier [11]. In order to achieve logic utilization up to 98% the relation between K and N has been experimentally obtained in [11] as follows:

$$I = (N + 1) \times \frac{K}{2} \quad (1)$$

By increasing either LUT size, K , or cluster size, N , the functionality of the logic block is enhanced. This can lead to a decrease in the total number of logic blocks needed to implement a circuit, but the size of the logic block increases with both K and N . The size of the cluster is quadratic in N [4]. In addition, and more importantly, the routing area depends on K and N . It has been experimentally shown that the best area-delay can be obtained by using a cluster size between 4 and 10 and LUT sizes of 4 to 6 [14]. Betz in [14] showed that the uniform distribution of pins on the logic block perimeter is better than top/bottom pin architecture in terms of routability and area.

The remaining three routability factors are algorithmic and relatively architecture independent. They all naturally depend on characteristics of the input circuit. These factors can be considered in technology mapping or packing algorithms. *Ratio of pins per net* indicates the density of high-fanout nets in the circuit. Routers spend the major part of their time for routing high-fanout nets. Hence the goal is to make this ratio low. *Ratio of used pins to total number of pins of the logic block* indicates the traffic in and out around a logic block. A block with high connection traffic is likely to create a congested area on the chip. Similarly high connectivity between logic blocks can lead to congestion. *Number of exposed nets* is also closely related to routability. As the number of exposed nets increases it is more difficult for the router to complete the routing. In a cluster-based FPGA, local routing inside clusters is not a concern compared to routing the external nets among clusters. In the technology mapping and packing stages the number of external nets can be reduced by inserting as many nets as possible inside the clusters. Utilization is another important issue in packing stage. Each routability metric has an impact on utilization as well. For instance, routability factor, *ratio of used pin per cell*, has the tendency to reduce the traffic around each block. This in turn leads to a decrease in logic utilization of a FPGA.

DeHon addresses the effect of depopulation of logic clusters on routability [10]. With less utilization of logic in CLBs, there would be lesser number of interconnection between CLBs. However, high depopulation would result in increase on the number of CLBs, longer connections and hence more routing congestion (which degrade the routability). In [9] the effort is to increase the routability by depopulating the CLBs. Based on the connectivity of the circuit obtained by Rent's rule, depopulation is applied. Although it is a good idea to consider routability based on the topology of the circuit, the weight of depopulation should vary during clustering as opposed to the static scheme used in the aforementioned work. We will discuss the

depopulation issue in the following sections. In this paper, we are focusing on algorithmic routability factors and we will actually show that they can affect routability and utilization significantly.

3.3 Problem Formulation

The input to packing stage for cluster-based FPGAs is a set of basic logic blocks. Maximizing utilization and routability are two main objectives. The problem can be formulated as follows:

Given a set of basic blocks, $B = \{b_1, b_2, \dots, b_n\}$ and set of CLBs each including K basic blocks, $C = \{c_1, c_2, \dots, c_m\}$, pack basic blocks into CLBs subject to the following constraints:

$$IN(c_i) \leq I \quad i = 1, \dots, m, \tag{2}$$

$$c_i \cap c_j = \emptyset \quad i, j = 1, \dots, m, i \neq j, \tag{3}$$

$$b_i \in \bigcup_{j=1}^m c_j \quad i = 1, \dots, n, j = 1, \dots, m. \tag{4}$$

The objective is to maximize routability and utilization (minimize $|C|$). The constraint in Equation 2 represents the limit on the number of input pins per CLB. The constraints in Equation 3 and Equation 4 show the packing definition, i.e. each block b_i should be assigned exactly to one CLB.

The packing problem as defined above is a clustering problem¹. First, we transform the problem to graph clustering problems. The input to packing problem, set of basic blocks and inter-blocks connection can be represented as an undirected *connectivity graph* G . Each node v in G corresponds to a basic block. An edge in the graph shows the interconnection. The multi-terminal nets are hyper-edges in this graph. Clustering problem is solved on graph G . In our version of the clustering problem, the number of clusters is not given. However there are two constraints on each cluster: first, each cluster consists of a fixed number of resources and second, the number of incoming and outgoing edges to each cluster is limited. The latter conflicts with the complete usage of resources in each cluster. Basically the second constraint is a limit on the density of connectivity between clusters, i.e. helps routability to some extent.

Routability is not a well-defined metric. As mentioned earlier in this section, it depends on a variety of factors. Reducing external nets and distributing the interconnection between the clusters homogeneously are two intuitive objectives. Achieving these objectives will lead to the ultimate goal of realizing interconnections of a circuit successfully while using minimum number of routing resources.

4 RPACK:ROUTABILITY-DRIVEN PACKING ALGORITHM

In this section, we explain how routability is incorporated into clustering process. We discuss the clustering heuristic based on our routability model.

¹In this paper, the terms “packing” and “clustering” have been used interchangeably.

4.1 Sequential Clustering Approach

Utilization and routability are two main issues in packing problem. Better utilization leads to a decrease in the number of clusters and routability is closely related to the amount of inter-cluster connection. Incorporating balancing, i.e. homogeneous utilization of the fixed amount of resources in each cluster, into objective may lead to a decrease in overall utilization. Constraints on the number of incoming and outgoing edges to each cluster limits the amount of interconnection among clusters. In other words, the constraint on number of edges around each cluster along with the fact that there are a fixed number of resources in each cluster has impact in balancing clusters. Utilization is high when most of the resources of clusters are used. Routability increases by reducing the complexity and the number of the interconnections between clusters. By covering as many edges as possible inside the clusters the connectivity among the clusters can be simplified. Therefore, maximizing routability can be viewed as a local objective in clustering.

According to the constraints, utilization, and routability issues defined for packing problem, a bottom-up approach in clustering is more applicable. Other clustering methods such as matching or top-down approaches which aim balancing, are not very suitable for packing in cluster-based FPGAs. Matching-based methods can be most suitable for packing blocks when the size of each cluster is 2 [23]. Also handling the constraints during optimization is not easy in such methods. Hence we take a greedy(bottom-up) approach for clustering. First a seed for a cluster is chosen. Then blocks are sequentially assigned to the cluster until the cluster is full or the limit on the number of input pins does not allow any more blocks to be assigned in the current cluster. Another seed is chosen for the next cluster. This continues until all the blocks are assigned to a cluster, while all the constraints are met. In [5, 6, 9, 8] same approach is used to pack the LUTs into clusters.

When a cluster is being constructed, there is always a problem on choosing the best candidate to be added to the cluster. This decision should lead to a locally optimal (or close to optimal) and feasible solution for the sub-problem. In this method, the feasibility of the solution is initially ignored and the best solution (local solution) is found. After that the feasibility is checked. If it is a legal solution for the sub-problem, it is added to the global solution. Otherwise the next best candidate is considered. Therefore the optimization problem is always followed by a decision problem for feasibility.

In such a greedy clustering method, each block selected to be inserted in a cluster is the local optimum choice under the local objective e.g. maximizing the number of edges shared with the cluster. Basically this is done by assigning a weight on any edge between any of the candidate blocks and cluster. Each time a candidate with best weight(gain) is chosen. In different approaches this weight is called connectivity [16], attraction [6, 5, 14], closeness [21], or edge separability [25], etc. We first discuss the important factors that have to be considered in assigning weights to edges. The weight of an edge should include factors affecting both utilization and routability (connectivity complexity). In [21], closeness is based on edge covering and balancing size of clusters. In this method type of connectivity is not considered. In [25] the weight of each edge is related to separability of the edge, which is computed based on min-cut between two ends of

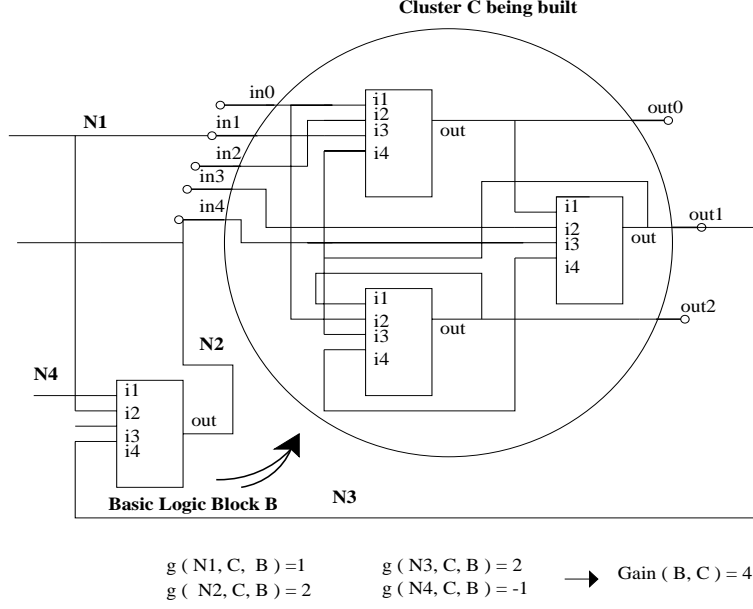


Figure 3: A Cluster Being Constructed and a Candidate Block, $K = 4, I = 5$

the the edge. Based on weight of the edges, clusters are merged in bottom-up manner while balancing the clusters. The utilization issue in FPGA packing problem cannot be efficiently handled by this method. In [5], attraction of an object while being added to a cluster is the number of nets the object and cluster share. The value of the attraction function reflects the number of edges being covered without considering routability issues for each net individually. Note that the greedy method of clustering as mentioned earlier, supports the utilization of the clusters. Therefore weight of edges do not need to include factors related to balancing.

Assume that the gain computed for a candidate block B , is the number of inputs and outputs it has in common with current cluster C [4] as defined in Equation 5.

$$RoutabilityGain(B) = |Nets(B) \cap Nets(C)| \quad (5)$$

In Figure 3, a candidate basic block B and cluster C being constructed are shown. According to the definition of the gain function in Equation 5, nets N_1 , N_2 , and, N_3 have the same contribution to the gain of block B to be added to the cluster C . If these nets are observed closely, their contributions to routability gain of the block are slightly different. Block B has three common nets with cluster C : N_1 , N_2 and N_3 . An input pin of the net N_1 is inside the cluster C . By adding the block B to the cluster, another input terminal of the net N_1 would be inside the cluster. This leads to a decrease in the number of the terminals of the multi-terminal net N_1 . The gain obtained by moving B to cluster C corresponding to net N_1 using Equation 5 is 1. In fact, the gain function originates from the third routability factor discussed in Section 3, i.e. reducing pins per net.

The driving pin of net N_2 is the output of the logic block B . There is an input pin of net N_2 inside the

pin in $B \rightarrow$ pin in C	status	in-pin gain	edge gain	output congestion	total gain
out \rightarrow in	-	1	1	0	2
out \rightarrow in	all in pins in C	1	1	1	3
in \rightarrow in in \rightarrow out	-	0	1	0	1
in \rightarrow out	rest of pins already in C	0	1	1	2
out	new net	0	0	0	0
in	new net	-1	0	0	-1

Table 1: Routability Gain of a Candidate Block According to a Single Net

cluster C . If the output terminal of a net is inside the cluster, internal connections can be used to connect the input pins of the net located inside the cluster. In such a case, there is no need to use an input pin of the cluster to connect the net N_2 to other terminals of the net outside the cluster, since an output pin of the cluster can be used for external interconnection. Therefore, the contribution of net N_2 to block gain is more than just covering an edge of a multi terminal net. Actually, by adding block B to cluster C , an input pin of the cluster gets free and can be used for another net connection. In Table 1 this is defined as *in-pin gain*. This increases the probability of acceptance of adding the block to the cluster. In other words, the probability of violating the input constraints of the cluster decreases. Note that each block output pin is accessible from outside and there is no sharing among the output pins. Therefore there would be no output pin constraints for the clusters. hence, saving on output pins does not bring any gain except in one case. Suppose all the input pins of a net are already inside the cluster and the logic block being added to a cluster contains the output pin of the net. Net N_3 in Figure 3 is an example of such a case. The output pin of the cluster corresponding to the block driving the net N_3 cannot be used by other blocks. This means that there would be no connection from outside to this pin since all the terminals of the corresponding net are located inside the cluster. Therefore the number of external connections of the cluster, defined as *output congestion gain* in Table 1, decreases. This yields less congestion among the clusters. In other words, it reduces the number of used pins of a cluster which is the fourth routability factor.

Net N_4 has no pin in the cluster. The gain from moving logic block B to the cluster would be zero according to the gain function above. However, not only no edge from N_4 would be covered, but also one input pin of the cluster would be used for N_4 . So the gain of moving logic block B to the cluster due to N_4 in terms of used pins per cluster is -1 . This means N_4 has a degrading effect on the routability according

to the fourth routability factor.

As explained above, by considering just the number of shared inputs and outputs as in Equation 5, the packing algorithm cannot differentiate among the candidate blocks which have different impacts on routability. All possible cases yielding different total gains are presented in Table 1 for one net connected to a candidate block.

By incorporating the other routability factors, the gain for each logic block B going into cluster C can be computed as the weighted combination of different routability factors as follows:

$$\begin{aligned} Gain(B, C) &= f(Nets(B), Nets(C)) \\ &= \sum_{i \in Nets(B)} g(i, Nets(C), B), \end{aligned} \tag{6}$$

where

$$g(i, C, B) = \begin{cases} 1 + a \times f_{in}(P(i, B), P(i, C)) \\ \quad + b \times f_o(P(i, B), P(i, C)) & i \in Nets(C) \\ -1 \times c \times T(i, B) & \text{otherwise} \end{cases}$$

$f_{in}(P(i, B), P(i, C))$ is defined as the gain obtained in input pins of cluster C as defined in Table 1. Similarly $f_o(P(i, B), P(i, C))$ is the gain obtained in output congestion. The additional gain of value 1 to the sum of these two gain terms corresponds to the edge gain. $T(i, B)$ returns the type of the pin of Net i connected to basic block B . It returns 0 if the pin is an output pin, and 1 otherwise. $P(i, B)$ is the set of all pins of Net i that are on block B . $P(i, C)$ is the set of pins of Net i connected to cluster C . $Nets(C)$ is the set of nets connected to cluster C . a , b , and c are the weights for different components of the function.

Inserting a whole multi-terminal net in one cluster is practically impossible. In most of the cases the best we can do is to eliminate two-terminal nets. Therefore, reducing an edge from a multi-terminal net should not be considered equivalent to reducing an edge by inserting a two-terminal net inside a cluster. The average gain that a block can take from an n -terminal net i connected to one of its pins, depending on type of the net can be estimated from Table 1 as follows:

$$Gain_{avg}(i, n) = \begin{cases} 2 & n = 2 \\ \frac{5}{3} \frac{1}{n} + \frac{2}{3} \frac{n-1}{n} & n > 2 \end{cases} \tag{7}$$

According to Equation 7, the average gain obtained from a two-terminal net is the highest. This implies that the algorithm gives priority to pushing a two-terminal net entirely inside a cluster as compared to reducing a pin of a multi-terminal net. This leads to a decrease in number of exposed nets satisfying the last routability factor.

When the net connected to a block is a multi-terminal net, the gain associated with the multi-terminal net is computed for each block containing a terminal of the net. Therefore, each edge of a net can have different impact on their corresponding blocks when a cluster is being constructed. How many and what type of a terminal of the net do already exist in the cluster? What type of terminal of the net does the candidate block have? Answers to these questions for each net connected to the candidate block determine the gain of the block. Therefore we conclude that in the bottom-up clustering gain(weight) of each edge should be assigned dynamically according to individual situations.

In the next sub-sections, we explain our method of packing the basic blocks inside the clusters based on the routability gain function mentioned. We also analyze timing-driven clustering algorithm in terms of routability based on routability gain function (Equations 6 and 7). According to this analysis, we integrate routability factors into timing-driven clustering.

4.2 RPack Algorithm

The input to our packing algorithm, is a list of LUTs, registers, and connections among the resources. In RPack, similar to VPack [4, 14], in the first stage, a LUT and a register are packed into a basic logic block when possible. After that, the blocks are packed into clusters using a greedy heuristic. Clusters are constructed sequentially. First, the seed is chosen from the unclustered basic blocks. The criteria is to choose the block with the most used inputs as mentioned in [4, 14]. After choosing a seed for a cluster, the logic block that gives the highest gain is selected to be added to the current cluster provided that it is a legal choice. This means that the number of external inputs do not exceed the number of input pins of the cluster. The algorithm continues adding blocks into one cluster until the cluster is full or no more legal choices can be found. Similarly new clusters are constructed until all the blocks are packed into clusters. We propose RPack, a routability-driven packing algorithm based on routability factors described in previous sub-section. RPack is developed on top of VPack. The difference between the two approaches is in the definition of gain function. VPack uses the function defined in Equation 5 while RPack uses the gain function in Equation 6. The pseudo code of our approach is shown in Figure 4.

The complexity of RPack Algorithm is $O(I^2 \times M^2)$, where M is the number of clusters. Finding the seed for each cluster takes $O(M^2)$ time using a priority queue to store the candidate nodes, where M is the number of nodes (basic blocks). When a node v is inserted to a cluster, only the gain of the neighbors of candidate nodes (Candidate nodes are those who have not been assigned to any node so far) need to be updated. The number of neighbors is equal to the edge degree of the current node, i.e. $deg(v)$. When a neighbor is visited, the type and status of the edges connected to the neighbor are checked which takes $O(deg(v))$. Note that when each neighbor node is visited the edges that belong to the same hyper-edge (multi-terminal net) is counted once. However, when a block is being added to a cluster, the number of neighbors are all the nodes connected to the node by any edge, i.e. $deg(v)$. By amortized analysis, it is observed that the gain of a node is updated at most once associated with any connection between the node and the neighbors. Therefore the

Input: Netlist of LUTs and Registers

N = Cluster Size

K = LUT Size

I = Inputs per Cluster

Output: List of Logic Clusters

```
1 Pack LUTs and Registers together into Basic Blocks
2 while Unclustered Basic Blocks available
3     Find Seed for new Cluster
4     while Cluster is not full
5         Update gains of unclustered Basic Blocks
        //Candidate blocks
6         Choose Basic Block with highest gain
        // Pick a candidate block
7         If Candidate is NOT feasible
8             then Go to Step 6
9         Else
10            Remove block from
            unclustered blocks list
11            Add block to current Cluster
12     end while
13 end while
```

Figure 4: RPack Pseudo-code for Packing Algorithm

total clustering process takes $O(\sum_{v_i \in V(G)} deg(v_i)) = O(2 \times |E|)$, where E is the edge set of connectivity graph G . Also $E \leq \sum_{i \in Net} (n_i)^2 \leq (\sum_{i \in Net} n_i)^2 \leq P^2$, where n_i is the number of the terminals of the net i and P is the total number of pins for all clusters, which is $I \times M$. Based on this analysis, the complexity of the algorithm can be expressed as $O(I^2 \times M^2)$.

4.3 t-RPack: Timing-Driven RPack

By clustering the LUTs in coarser CLBs, the complexity of interconnection between the CLBs is reduced. Hence fewer number of routing resources is required. Another benefit of clusters is the fast interconnection inside the clusters. Those connections being packed inside the clusters, use the hard-wired interconnect resources of CLBs. This leads to better performance. In packing, both objectives should be pursued. In this paper, our focus is mostly on routability. In this section we discuss how routability is realized when timing is added into packing algorithm and based on our routability function we propose timing-driven RPack.

After packing, a subset of the netlist is routed inside the clusters without passing through switched routing resources. By inserting the interconnection along the critical path of the circuit inside clusters delay can be improved. As a result, in timing-driven clustering the priority is given to timing critical connections to be inserted inside the clusters.

In sequential bottom-up clustering approach, a seed for a cluster would be the most critical block. The blocks are added to the cluster based on criticality. In addition to timing, routability has to be considered in clustering to avoid the routing congestion, which is a bottleneck in current FPGAs. However, first we should study the impact of timing-based clustering on routability when choosing the seed and defining the gain function based on criticality. In Section 3 the routability factors are described. Based on that, our routability gain function is defined in Equation 6. Using this model, we can explain the routability issues in timing-driven clustering. After analyzing the approach, we would be able to improve the routability more accurately. This is where analysis and theory guide the heuristics.

The criticality of the blocks are defined by their slack. Connections along the critical paths have high criticality value. Therefore, clustering based on timing is similar to path-based clustering. Each path is a chain of output-to-input pin-to-pin connections between a set of blocks (See Figure 5). According to routability gain function (Table 1), the output-to-input connection has a high routability gain. When a connection is marked to be critical, it means that there is a long chain of input-output connectivity from this point to the rest of the design. This implies a prediction of high routability gain in later stages while constructing the cluster. After a highly critical connection is added to a cluster, more input-output connections would be added to the cluster. In other words, criticality of a connection shows the depth of input-output connections from the current connection to the rest of the circuit. By inserting an edge of the net on the critical path, timing-driven packing exploits the routability obtained by inserting output and input pin of a net inside a cluster, hence releasing an input pin of the cluster. As explained above, our routability model can express the routability impact of timing-based clustering.

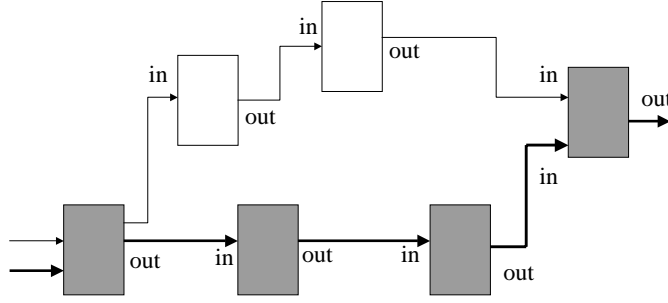


Figure 5: Routability and Slack Computation.

The two terms of routability factors are inherently satisfied in criticality-based analysis and slack computation for critical connections. Other factors should be considered. In addition, routability for non-critical nets should be taken into account during clustering. Therefore, we define the gain function as a linear combination of criticality and routability of a connection. We use the same criticality function used in t-VPack[6]. The routability component is routability gain function defined in [5]. Equation 8 shows the gain function used in timing-driven RPack.

$$TotalGain(B) = \alpha \times Criticality(B) + (1 - \alpha) \times \frac{RoutabilityGain(B)}{DepopulationFactor} \quad (8)$$

Another important issue is scaling the routability component in Equation 8. When a cluster is just being constructed, there are many available un-used pins of the cluster. In this stage, the cluster desires to absorb as many connection as possible. In later stages, when most of the pins are used, routability is more restricted and the used pins around the clusters create congestion around the block. In this case, depopulation can help improve the routability. In addition, when more blocks are added to the cluster, the probability of getting higher gain in the later stages is increased due to the higher probability of existence of shared nets among the blocks. This does not imply the higher routability due to higher connection traffic around the cluster. Therefore, scaling is required. In order to achieve this, the routability function value is scaled each time a block is added to the cluster. The depopulation factor increases during the construction of a cluster. The depopulation factor is defined as follows :

$$DepopulationFactor = UsedPin(B) + UsedPin(C), \quad (9)$$

$UsedPin(B)$ and $UsedPin(C)$ return the number of used pins of block B and cluster C , respectively. We need to mention that in t-VPack[6], the total gain function is a function of routability and criticality as well.

However the routability factor is same as the gain function used in VPack which is not a comprehensive routability function. Also, the routability is scaled by the number of pins of a LUT, i.e 5 for 4-input LUT. This normalization remains constant during the clustering. According to our discussion above this cannot reflect routability gain correctly.

With analysis and more accurate modeling of routability, we are able to study the behavior of different methods of clustering in terms of routability and improve the approaches by having additional components considering other routability factors. Our analysis in this section shows that timing and routability correlate very strongly. Satisfying timing improves routability in some aspects. That is why timing-driven clustering outperforms a routability-driven packing. Our experimental results in the next section supports our claim as well.

5 EXPERIMENTAL RESULTS

In previous sections we claimed that considering routability factors while packing logic into CLBs has significant impact in routing results and netlist complexity. In this section, we show a set of experimental results supporting our claim.

We have used the greedy clustering approach proposed in VPack and t-VPack. RPack is implemented on top of the clustering algorithms in V-Pack and t-RPack is implemented on top of t-VPack.

The first set of our experiments compares RPack and VPack. We ran the 20 largest MCNC benchmarks [28] on VPack and RPack. The blif input format of each benchmark is obtained by SIS [27] logic minimization and FlowMap [18] technology mapper. The results presented in Table 2 show that our method successfully decreased the number of exposed nets. RPack and VPack use similar number of clusters such that the array size resulting from both approaches for almost all benchmarks is same. Even in one case (benchmark “alu4”) RPack yielded smaller array size. The array size of each benchmark is reported in Table 2. In accordance with average gain estimated in Section 4, the results show that the major portion of the decrease in the number of the exposed nets is due to decrease in the number of two-terminal nets. In conclusion, reducing the number of output pins is strongly related to reducing the number of exposed nets.

We also observed the congestion around each cluster. We counted the number of exposed nets each cluster is connected to. Figure 6 shows the connectivity of the clusters resulted from VPack and RPack for benchmark *bigkey*. The size of cluster is 8 and number of input pins per cluster is 18. The vertical axis shows the number of clusters for each number of pins used per cluster shown on the horizontal axis. The plot shows that the clusters obtained from RPack have less traffic around. In Figure 7, the result for benchmark *elliptic* is shown as well. As shown in the plot, the connectivity obtained from RPack is more smoothly distributed compared to the one resulted from VPack. In these two plots the type of interconnection is not reflected. The number of terminals of the nets also affects routability.

In order to verify that our method meets the objective of improving routability, we synthesized the

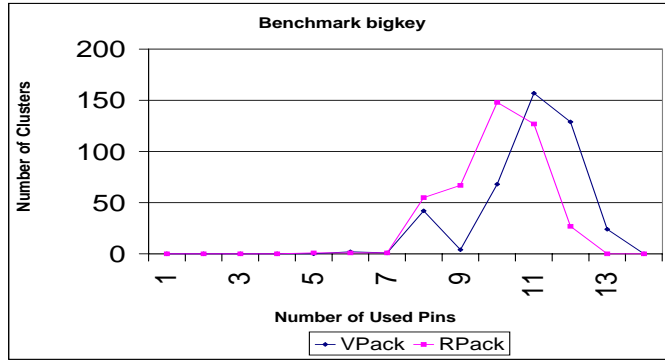


Figure 6: Comparison of RPack and VPack in cluster characteristics in *bigkey*.

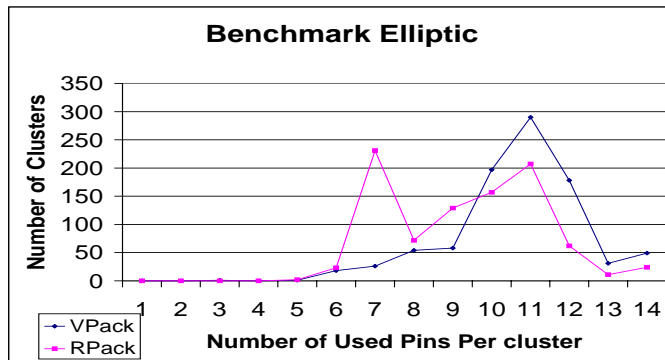


Figure 7: Comparison of VPack and RPack in cluster characteristics in *elliptic*.

benchmark circuits through the complete CAD flow to obtain the routing area. We used VPR [4] to place and route the benchmarks. The routing architecture that we used employs only the single segment wires leading to better routability. *Subset* switch type, in which each track in a channel can be connected to the same track number of the neighboring channels, is used. In addition we have set the fraction of the tracks of each channel to which each logic block input and output pins connect to 1.

Table 2 summarizes our results after placement and routing of the benchmarks. As shown in Table 2 RPack is able to improve the routing area by decreasing the number of tracks significantly. The average improvement we obtained is 16.5 %. The number of routing tracks required in each channel is a reliable metric for routing area, since a smaller number of routing tracks does not only mean saving wiring area, but also decreasing the size of the routing switches drastically. Routing area is related to the square of the number of tracks per channel. The improvement in routing area is 27% over VPack on an average. Such an improvement in routing area decreases the total chip area significantly, since routing area is typically a large percentage of the total area.

The significant difference between the two routability-driven methods of VPack and RPack implies that each routability factor can affect the routing results significantly. According to the constraint on number of pins per CLB, fixed routing resources, and fixed number of LUTs in each CLB, we considered different routability components in the gain function used in RPack. The results support our claim that routability is an important objective in clustering and it results in a better distribution of interconnection among CLBs.

In the next set of experiments, timing-driven RPack is compared with t-VPack, in order to observe the impact of routability in timing-driven packing. It is not a correct comparison if RPack is compared with t-VPack. As mentioned in previous section, timing based clustering inherently has effective impact on routability itself. Routability and speed, both benefits of cluster-based FPGAs, are realized by timing-based packing algorithm. Previous work shows that t-VPack performs better in terms of routability compared to VPack.

The routing results, critical path delay and number of exposed nets using both packing methods, t-RPack and t-VPack, are reported in Table 3. t-RPack uses our routability gain function as described in Section 4. In order to observe the effect of routability gain function, depopulation factor is ignored. The results show that considering other routability factors and more accurate routability gain for non-critical nets can improve the routing area by 5.2%. The delay is improved by 3%. The reason is that the weight for input-output connection is high in both timing and routability component for critical nets. We can observe that delay has been improved in most of cases. In another set of experiments, we added the depopulation factor to control the routability versus timing. The results are shown in Table 4. The routing area is improved by 12% while the critical path delay is same on average. This implies that depopulation helps to obtain a more distributed connectivity between the clusters.

The experimental results show that different routability factors have significant impact on routing results. Timing-driven packing has strong correlation with some of routability factors for FPGAs. Integrating

circuit	Array Size	Number of Exposed Nets		Number of Tracks		
		VPack	RPack	VPack	RPack	%
alu4	14 × 14	1296	985	42	35	27.4
apex2	16 × 16	1626	1288	43	38	11.6
apex4	13 × 13	1037	868	40	39	2.5
bigkey	27 × 27	1622	1060	28	14	50
clma	33 × 33	7139	5585	59	51	13.6
des	32 × 32	1601	1339	24	19	20.8
diffeq	14 × 14	1280	895	25	20	20
dsip	27 × 27	1590	1219	29	25	13.8
elliptic	22 × 22	3244	2300	47	35	25.5
ex1010	25 × 25	3799	3064	48	44	8.3
ex5p	12 × 12	950	754	39	38	2.6
frisc	22 × 22	2955	1983	43	36	16.3
misex3	14 × 14	1101	876	36	33	8.3
pdc	25 × 25	3813	3011	63	57	9.5
s298	16 × 16	1711	1330	37	30	18.9
s38417	29 × 29	4921	3921	36	28	22.2
s38584.1	29 × 29	4649	3556	34	27	20.6
seq	15 × 15	1496	1166	43	38	11.6
spla	22 × 22	3031	2336	60	49	18.3
tseng	12 × 12	979	764	26	24	7.7
Average	-	2492	1913	40.1	34.35	16.5

Table 2: Logic Size, Number of Exposed Nets, and Number of Routing Tracks: RPack vs. VPack.
N=8, K=4, I=18, Routability-driven Router

circuit	Array Size	Number of Exposed Nets		Number of Tracks		Delay ($\times 10^{-9}$ s)	
		t-VPack	t-RPack	t-VPack	t-RPack	t-VPack	t-RPack
alu4	14 \times 14	804	737	32	29	30	22.8
apex2	16 \times 16	1249	1087	40	36	28.5	26.6
apex4	13 \times 13	868	798	43	41	23.1	22.5
bigkey	27 \times 27	1040	754	18	15	13.2	12.2
clma	33 \times 33	5307	4956	55	56	50.6	47.4
des	32 \times 32	1214	1109	18	17	27.2	25.7
diffeq	14 \times 14	1033	926	22	22	31.3	35
dsip	27 \times 27	762	767	14	17	15.8	12.4
elliptic	22 \times 22	2247	1960	42	36	40	43
ex1010	25 \times 25	3110	2894	47	46	34	31.6
ex5p	12 \times 12	767	724	39	42	27.5	24.9
frisc	22 \times 22	2048	1732	44	44	57.5	58.2
misex3	14 \times 14	840	781	35	34	23.9	23
pdv	25 \times 25	2627	2466	62	60	35.3	34.2
s298	16 \times 16	767	725	24	24	49.3	46.3
s38417	29 \times 29	4423	3881	31	32	36.1	32.9
s38584	29 \times 29	4183	3671	35	32	28.3	26.6
seq	15 \times 15	1055	938	39	36	23.9	21.7
spla	22 \times 22	2099	1984	49	51	31.4	29.8
tseng	12 \times 12	801	774	25	25	32.7	31.1
Average	-	1810	1683.2	35.7	34.75	32	30.4

Table 3: Logic Size, Number of Exposed Nets, Number of Routing Tracks, and Critical Path: t-RPack (without depopulation) vs. t-VPack. N=8, K=4, I=18, Routing Algorithm: timing-driven

remaining routability factors for non-critical nets can improve the routability of timing-driven packing algorithms.

6 CONCLUSIONS AND FUTURE WORK

In this paper we addressed routability issues and their impact on performance and routing area. A routability-driven packing method for cluster-based FPGAs is proposed. Our method is able to improve the routability by decreasing the number of required tracks in the FPGA routing channels. This improvement was achieved by incorporating several routability factors in our packing algorithm. Based on our routability model, we analyzed the timing-driven packing. Criticality of a connection in terms of timing reflects the role of this connection in routability as well. We integrated our routability function into timing-based packing to improve the routability. We were able to decrease the routing area by 20% on average compared to existing packing methods.

Assigning better and more accurate weights for hyper-edges in connectivity graph for clustering is the future work we are focusing on. Incorporating the effect of number of terminals by dynamically assigning weights for edges can improve the results in terms of routability. Routability issues should be incorporated into placement and routing tools as well.

circuit	Array Size	Number of Exposed Nets		Number of Tracks		Delay ($\times 10^{-9}$ s)	
		t-VPack	t-VRPack	t-VPack	t-RPack	t-VPack	t-RPack
alu4	14 \times 14	804	724	32	27	30	24.6
apex2	16 \times 16	1249	1086	40	39	28.5	26.6
apex4	13 \times 13	868	740	43	39	23.1	22.5
bigkey	27 \times 27	1040	747	18	14	13.2	12.2
clma	33 \times 33	5307	5003	55	53	50.6	52.6
des	32 \times 32	1214	1111	18	15	27.2	26
diffeq	14 \times 14	1033	912	22	23	31.3	36
dsip	27 \times 27	762	767	14	14	15.8	14
elliptic	22 \times 22	2247	1878	42	39	40	42
ex1010	25 \times 25	3110	2856	47	43	34	33
ex5p	12 \times 12	767	743	39	40	27.5	26.7
frisc	22 \times 22	2048	1798	44	44	57.5	58.4
misex3	14 \times 14	840	766	35	34	23.9	23.5
pdv	25 \times 25	2627	2472	62	60	35.3	36
s298	16 \times 16	767	740	24	24	49.3	49.2
s38417	29 \times 29	4423	3985	31	29	36.1	37
s38584	29 \times 29	4183	3794	35	31	28.3	27.8
seq	15 \times 15	1055	944	39	36	23.9	24.7
spla	22 \times 22	2099	1954	49	49	31.4	32
tseng	12 \times 12	801	584	25	21	32.7	32
Average	-	1810	1680.2	35.7	33.6	32	32

Table 4: Logic Size, Number of Exposed Nets, Number of Routing Tracks, and Critical Path: t-RPack(with population factor) vs. t-VPack. N=8, K=4, I=18, Routing Algorithm: timing-driven

References

- [1] S. Brown, J. Rose, "Architecture of FPGAs and CPLDs: A Tutorial". *IEEE DESIGN AND TEST OF COMPUTERS*, Vol. 13: (2), pp. 42-57, SUM 1996.
- [2] M. R.Garey, D. S. Johnson, "Computers and Intractability, A guide to the Theory of NP-Completeness," *W. H.Freeman and Company*,1999.
- [3] D. T.Hochbaum, "Approximation Algorithms for NP-hard Problems" *PWS Publishing Company*,1995.
- [4] V. Betz, J. Rose, "VPR: A New Packing Placement and Routing Tool for FPGA Research," *Int. Workshop on Field-Programmable Logic and Application*, pp. 213-222, 1997.
- [5] V. Betz, J. Rose, "Cluster-based Logic Blocks for FPGAs: Area Efficiency vs. Input Sharing and Size," *IEEE Custom Integrated Circuits Conference*, pp. 551-554, 1997.
- [6] A. Marquardt, V. Betz, J. Rose, "Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density," *ACM/IEEE International Symposium on FPGAs*, Feb 1999.
- [7] R. J. Francis, J. Rose, Z. Vranesic, "Technology Mapping for Delay Optimization of Lookup Table-Based FPGAs". *MCNC Logic Synthesis Workshop*, 1991.
- [8] E. Bozorgzadeh, S. Ogreneci Memik, M. Sarrafzadeh, "RPack: Routability-driven Packing for Cluster-based FPGAs,". *Asia South Pacific Design Automation Conference*, Jan 2001.
- [9] A. Singh, G. Parthasarathy, M. Marek-Sadowska, "Interconnect Resource-Aware Placement for Hierarchical FPGAs,". *ACM/IEEE International Conference on CAD*, Nov. 2001.
- [10] A. DeHon, "Balancing Interconnect and Computation in a Reconfigurable Array". *ACM/IEEE International Symposium on FPGAs*, Feb. 1999.
- [11] E. Ahmed, J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density". *ACM/SIGDA International Symposium on FPGAs*, pp. 3-12, Feb 2000.
- [12] K. Chen, J. Cong, Y. Ding, A. Kahng, P. Trajmar, "DAG-Map: Graph-Based FPGA Technology Mapping for Delay Optimization". *IEEE Design and Test*, pp.7-20, Sep. 1992.
- [13] R. J. Francis, J. Rose, Z. Vranesic, "Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs ". *ACM/IEEE Design Automation Conference*, pp.613-619, 1991.
- [14] V. Betz, J. Rose, A. Marquardt, " Architecture and CAD for Deep-Submicron FPGAs," *Kluwer Academic Publishers*, 1999.
- [15] N. Bhat, D. Hill, "Routable Technology Mapping for FPGAs," *First International Workshop on FPGAs*, pp. 143-148, Feb. 1992.
- [16] D. M. Schuler, E. G. Ulrich. "Clustering and Linear Placement," *ACM/IEEE Design Automation Conference*, 1972.
- [17] M. Schlag, J. Kong, P. K. Chan, "Routability-Driven Technology Mapping for Lookup Table-Based FPGAs," *IEEE Transactions on CAD*, Vol. 13, pp.13-26, Jan 1994.

- [18] J. Cong, Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Design," *IEEE Trans. on Computer-Aided Design*, Vol.13, No.1, pp. 1-12, Jan. 1994.
- [19] J. Cong, Y. Ding, "On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping," *IEEE Trans. on VLSI Systems*, Vol.2, pp.137-148, June 1994.
- [20] N. S. Woo, "A Heuristic Methods for FPGA Technology Mapping Based on the Edge Visibility," *ACM/IEEE Design Automation Conference*, pp. 248-251, June 1991.
- [21] H. Shin, C. Kim, "A Simple Yet Effective Technique for Partitioning," *IEEE Transaction on VLSI*, pp. 380-386, September 1993.
- [22] J. Cong, Y. Ding, T. Gao, K.C. Chen, "LUT-Based FPGA Technology Mapping under Arbitrary Net-Delay Model". *Computers and Graphics*, Vol.18, pp.507-516, 1994.
- [23] J. Cong, J. Peck, Y. Ding, "RASP : A General Logic Synthesis System for SRAM-based FPGAs,". *ACM Symposium on FPGAs*, pp.137-143, 1996.
- [24] K. Karplus, "XMap: A technology Mapper for Table-lookup Field-Programmable Gate Arrays". *ACM/IEEE Design Automation Conference*, pp.240-243, 1991.
- [25] J. Cong, S. K. Lim, "Edge Separability Based Circuit Clustering with Application to Circuit Partitioning". *Asia South Pacific Design Automation Conference*, pp. 429-434, Jan 2000.
- [26] C. J. Alpert, A. B. Khang, "Recent Directions in Netlist Partitioning: A survey,". *The VLSI Journal* 19, pp. 1-81, 1995.
- [27] E. M. Sentovich, et al, "SIS: A System for Sequential Analysis," *Tech. Report No. UCD/ERL M92/41* , University of California, Berkeley, 1992.
- [28] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0," *Tech. Report, Microelectronics Center of North Carolina*, 1991.