Congestion-Driven Re-Clustering CAD Flow for Low-Cost FPGAs

by

Darius Chiu

B.Sc., The University of British Columbia, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The Faculty of Graduate Studies

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

September, 2009

© Darius Chiu 2009

Abstract

FPGA device area is dominated by the on-chip interconnect. For this reason, the amount of interconnect provided must be limited. This limit is usually imposed by designing an FPGA device family with a fixed channel width. CAD tools must meet this hard channel-width constraint for a circuit to be successfully mapped to a device from this family. Previous work has shown that if a design cannot be mapped to a device due to insufficient interconnect availability, it is possible to identify regions of high interconnect demand, and spread out or *depopulate* the logic in this area into surrounding regions. This is done by re-packing logic in the affected regions into an increased number of CLBs. This increases the effective amount of interconnect available to these high-demand areas. This methodology has been shown to significantly reduce channel width, at the expense of CLB count and runtime.

In this work, we extend this previous algorithm in two ways: we present novel region selection techniques to optimize the selection of which regions should be depopulated, and we introduce a local channel-width demand model which can used to more accurately determine the amount of white space insertion at each iteration. Together, these techniques lead to significant run-time improvements and reduce the area of the resulting FPGA implementations. We were able to improve runtime by a factor of up to 5.5 times while reducing area by up to 20% when compared to previous methods.

Table of Contents

Appendices

List of Tables

List of Figures

Glossary

Field-Programmable Gate Arrays (FPGA)	An integrated circuit device which are capable of being programed to implement any digital circuit
Look-up Table (LUT)	An element of a FPGA device which impliments any logical function of its inputs
Configurable Logic Blocks(CLBs)	An element of a FPGA device which comprises of a group of N BLEs which are interconnected with a fast local interconnect network
Basic Logic Elements(BLEs)	An element of a FPGA device which is comprised of a LUT and Flip-Flop
Aplication Specific Integrated Circuit (ASIC)	An integrated circuit device which is specifically designed and manufactured for a specific purpose
Computer-Aided Design (CAD)	Automated computer software tools used to aid the design and implimentation of systems
Minimum Routable Channel Width (MRCW)	The minimum number of tracks required for a given design to be routeable
Channel Width	The number of wiring tracks in each routing channel in the FPGA
Hardware Description Language (HDL)	A human readable language to express hardware designs
Microelectronics Corporation of North Carolina (MCNC) Circuits	A set of widely used acedemic benchmark circuits for FPGAs

Acknowledgments

My sincere thanks goes out to Dr. Guy Lemiuex and all of my fellow students and professors who have given me assistance and guidance throughout the course of my degree.

In particular, I would like to thank David Grant for the help he has provided in maintaining our server cluster. Without his help, I would not have been able to easily complete this thesis. In addition, I would also like to thank Dr. Steve Wilton and Usman Ahmed for helping me with my writing.

Finally I would like to thank my friends Andrew, Cindy, Chris, Dave, Johnny, Paul, and Patrick for the great times during the course of this degree.

Dedication

I dedicate this to my family who have patiently given their support to me throughout the years.

Chapter 1

Introduction

Field Programmable Gate Arrays are customizable devices capable of implementing a variety of digital logic applications. This is because FPGAs can be customized by configuring re-programmable logic blocks and routing fabric. In contrast, Application Specific Integrated Circuits (ASICs) offer an alternative to FPGAs. ASICs are custom-manufactured integrated circuits which are designed for a specific application. Because of the application-specific nature of ASICs, they can typically offer better speed, density, and power characteristics than FPGAs. However, ASICs also require large up-front manufacturing costs and do not provide the re-programmability offered by FPGAs. FPGA devices are purchased pre-manufactured from the vendor. Time-to-market can be reduced because development and testing can be performed immediately using actual FPGA devices. Thus, FPGAs are favorable in circumstances where field-programmability and time-to-market benefits outweigh its speed, density and power disadvantages. In addition, low volume applications may not justify the economics of the high up-front costs for ASIC manufacture. FPGAs offer designers an alternative option for low to medium volume applications.

As FPGAs increase in capacity and capability, it is common for manufacturers to provide separate low-cost and resource-rich families. For a similar logic capacity, low-cost families often have less embedded memory, embedded multipliers, and interconnect in the form of routing tracks in each channel. To target low-cost

device families, computer-aided design (CAD) tools must configure the FPGA device in a way which meets a hard channel-width constraint imposed by routing capacity limits.

Careful allocation of FPGA logic can help meet routing capacity constraints. Device logic in FPGAs are grouped as *clusters*, also known as configurable logic blocks (CLBs). The distribution of logic among CLBs can impact the number of routing tracks used around each CLB; the interconnect use around CLBs varies spatiality with placement. It is possible to distribute regions of high local interconnect demand by spreading logic over a larger number of CLBs. This increase in area allows the same amount of logic access to an increase in aggregate routing. Therefore, it is possible to spread logic in regions where there is high local interconnect demand, allowing CAD tools to meet hard channel-width constraints by increasing CLB usage.

In a traditional single-pass CAD flow such as VPR [3], a solution may not be found which meets the channel-width constraint of a low-cost device. When regions are found which are congested, we can distribute logic over a larger area through whitespace insertion. Whitespace is inserted in the form of empty logic elements; for example, we can impose a limit on how many basic logic elements (BLEs) are allowed to form a CLB. Those CLBs in regions with higher local interconnect requirements should contain less BLEs such that local interconnect requirements are less then or equal to the channel-width provided by the device. The processes of identifying and re-clustering CLBs to reduce logic capacity is termed depopulation.

Depopulation can occur in different ways. Single-pass clustering approaches, such as that presented in [26], can perform clustering in a way which results in clusters that are not full. These clustering approaches typically estimate the

routability of the circuit at the clustering stage and attempt to perform clustering to ease routability. However, without detailed information from placement or routing solutions, it is difficult for a clustering tool to determine the routing requirements of the circuit. This is especially true if our goal is to reduce local interconnect usage to meet channel-width constraints. In order to effectively depopulate only the areas of the device which are unroutable, we need to accurately identify and select which CLBs to depopulate. We also need to determine the amount of depopulation needed, which will require accurate prediction of post-routing interconnect demand after depopulation has occured.

In Un/DoPack [29], authors presented methods to iteratively perform whitespace insertion. Post-routing information is used by the CAD flow to determine which regions should be targeted for depopulation. Regions which cannot be routed are reclustered to a smaller cluster size. This cluster size is determined either by the size of the region, or relative to the routability of the region. To our knowledge, to date, only Un/DoPack is capable of meeting a user-specified channel-width constraint through the use of iterative depopulation.

1.1 Contributions

The main contribution of this work is to improve region selection and whitespace insertion of Un/DoPack through the use of congestion information. We improve region selection, and effectively allocate whitespace to reduce routing requirements to meet channel-width constraints, but at the same time reduce runtime of the CAD flow and CLB usage. Un/DoPack showed that congested areas in the placement can be identified and reduced, but lacked an interconnect demand model to determine the amount of whitespace to insert. We will show that the

use of a model-driven depopulation approach can result in both runtime and area improvements.

Primarily, we will be improving area and runtime by selecting multiple congested regions simultaneously, and depopulating each region according to local interconnect demand. The model will be used to select congested areas in the placement for depopulation; this model will also attempt to predict the local interconnect demand after depopulation has occurred. This information will then be used to determine an adequate amount of whitespace to insert.

To compare the runtime and area improvements, we will compare our work against the baseline version of Un/DoPack presented in [29] using the same benchmark circuits and channel-width constraints. The benchmark circuits, composed of smaller sub-circuits, are designed to simulate large system-on-chip circuits. These circuits exhibit the property where each sub-circuit may have different local interconnection requirements. Compilation using various channel-width constraints are performed to compare the various schemes presented in this thesis.

1.2 Thesis Organization

The remainder of the thesis is organized as follows: Chapter 2 provides an overview of modern FPGA technology, the state of current FPGA CAD technology, and previous work relating to the subject of this thesis; Chapter 3 describes the algorithms used and the experimental methodology; Chapter 4 compares the results of this work against previous work; Chapter 5 presents the conclusions of this work, contributions, and possible future work.

Chapter 2

Background

This chapter provides an overview of current FPGA technology, as well as modern methods used to implement circuit designs onto an FPGA. A description of each step of a generalized CAD flow is discussed, along with a survey of common tools used in each step. We then describe, in detail, a CAD flow which performs regional depopulation to alleviate routing congestion.

2.1 FPGA Architecture

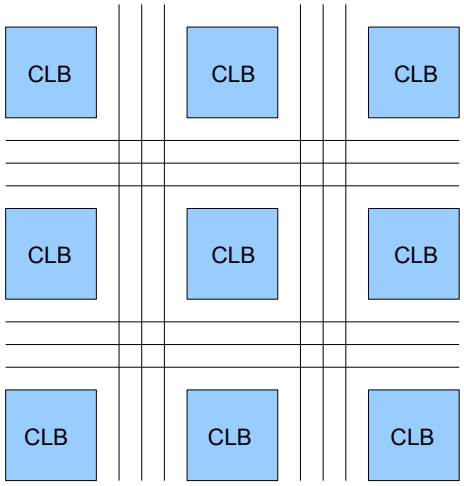


Figure 2.1: FPGA Logic and Routing Layout

Field-Programmable Gate Arrays (FPGAs) are integrated circuits which are capable of implementing any digital circuit. This is possible because FPGAs contain an array of programmable logic elements, which can communicate with each

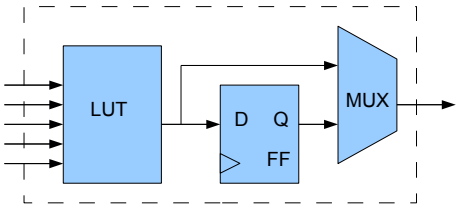


Figure 2.2: Basic Logic Element

other via a programmable routing fabric. This is shown in Figure 2.1 and is often referred to as the Island Style Architecture. Logic is arranged in a rectangular array and are surrounded by wires in the vertical and horizontal directions. The number of CLBs spanning the FPGA in the horizontal and vertical position is referred to as the *array size*. Surrounding the periphery of the FPGA, are input/output pads which allow the FPGA to connect with circuitry outside the FPGA.

FPGA logic elements are known as a *basic logic element* (BLE). The logic capacity of a FPGA device is commonly measured as the number of BLEs contained in the whole FPGA.

As shown in Figure 2.2, BLEs consist of a k-input *look-up table* (LUT) and a flip-flop. A k-input LUT, or k-LUT, can implement Boolean logic up to k inputs. The BLE can be used in combinational mode, where the output of the BLE is taken from the LUT, or sequential mode where the output is taken from the flip-flop. It has been shown that it is advantageous to combine multiple BLEs into clusters. These clusters contain a fast, local interconnect which connects constituent BLEs. Clusters connect to other clusters using an inter-cluster routing fabric. These clusters are known as *configurable logic blocks* (CLBs), shown in Figure 2.3. Previous work has shown that clustering creates various advantages including: the reduction of delay, reduction of interconnect usage, increase in

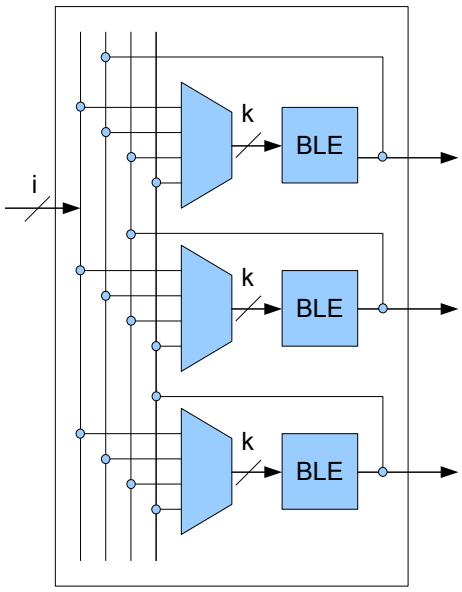


Figure 2.3: Generalized Configurable Logic Block (from [30])

device density, and most importantly a reduction in CAD runtime. The maximum number of BLEs contained within a single CLB is defined as N, and the number of inputs as i. [1] experimentally determined the relationship between the number of inputs required for a cluster as a function of the LUT size and cluster size. The number of inputs per cluster was determined to be $i = \frac{k}{2} \times (N + 1)$, where k is the LUT size and N is the number of BLEs per cluster. The authors in [1] determined that this relationship yielded a high utilization (98%) of the logic contained within the CLBs.

FPGA routing surrounds the CLBs. The routing fabric consists of three components: wires, connection blocks, and switch blocks. Wires occupy the vertical and horizontal channels adjacent to CLBs in the FPGA. In modern designs, wire segments may span multiple CLBs to allow efficient communication across longer distances. Because the number of wires occupying each channel are fixed at manufacture-time, the routing capacity of a FPGA device is limited by the num-

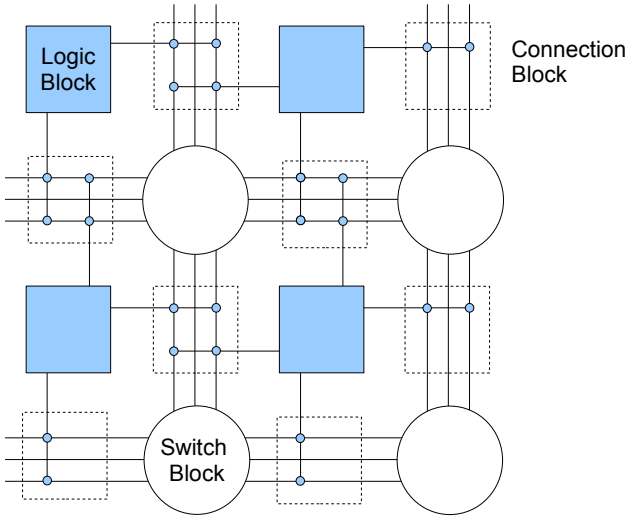


Figure 2.4: Routing

ber of wires contained in each channel across the device. The number of wires in each channel is referred to as the *channel width*. Channel width provides a major constraint for FPGA computer-aided design (CAD) tools since designs cannot be implemented if channel width constraints cannot be met; insufficient routing resources imply that connections between CLBs cannot be made. Furthermore, the channel width required may not be uniform across the device; some regions will require more interconnect than others. The occurrence of a small portion of the FPGA requiring more wiring than is offered by the FPGA architecture will prevent the entire design from being implemented. In this case, the circuit is said to be *unroutable*. The minimum channel width required for the circuit to be implemented is referred to as the *minimum routable channel width* (MRCW).

FPGA routing also contains connection blocks and switch blocks. *Connection blocks* allow CLBs to accept inputs from, or provide outputs to, nearby wires. *Switch blocks* allow signals to change direction by connecting vertical and horizontal routing wires together. This is shown in Figure 2.4

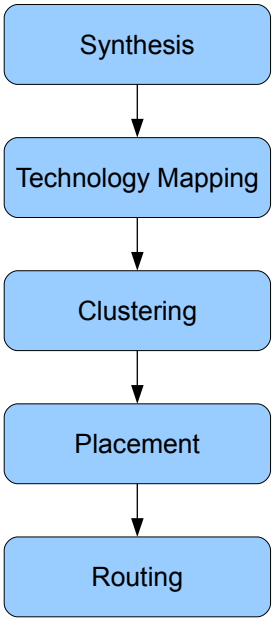


Figure 2.5: Typical FPGA CAD Flow

2.2 FPGA CAD Flow

Through the use of software tools, designers produce a bitstream used to program the FPGA device. Figure 2.5 shows the 5 typical steps, and is referred to as an FPGA CAD flow. These steps are: synthesis, technology mapping, clustering, placement, and routing. These steps are described in detail below.

2.2.1 Synthesis

A designer will typically express the design in a hardware description language (HDL), such as VHDL or Verilog. The task of the synthesis step is to translate the design into a gate-level representation. This representation is a network of Boolean logic gates and flip-flops.

2.2.2 Technology Mapping

The technology mapping step takes the output of the synthesis step and maps groups of logic into k-LUTs. At this stage, optimizations can be made to minimize logic usage, delay and power. For example, delay can be reduced by minimizing logic depth, which is the longest path of the circuit. Most notably, FlowMap [8] was able to produce a depth optimal-solution in polynomial complexity time. Other technology mapping algorithms are presented in [9], [10], [11] and [15].

2.2.3 Clustering

The clustering step packs LUTs and flip-flops into BLEs and combines multiple BLEs into CLBs to reduce delay and routing resource usage of the circuit. Each CLB contains a fast local interconnect structure which allows constituent BLEs to communicate with each other. This intra-cluster routing is much faster than inter-cluster routing in general.

The simplest clustering approaches are based on greedy selection. Known as the bottom-up approach, individual CLBs are built by first choosing a seed BLE. Subsequent blocks are added to the CLB based on their relationship to the seed. This occurs until CLB capacity constraints are met. Various greedy clustering algorithms exist, each with distinct goals. Examples include: Vpack [2], which attempts to minimize total number of inputs per cluster; T-VPack [3], which tries to reduce the number of intercluster nets on the critical path; RPack [4], which attempts to reduce the routing effort of the circuit by enclosing intercluster nets into clusters; an intrinsic shortest-path length based clustering method [23], which attempts to reduce post-placement wirelength; and iRAC [26] which attempts to minimize routing channel width by completely enclosing low-fanout nets within a

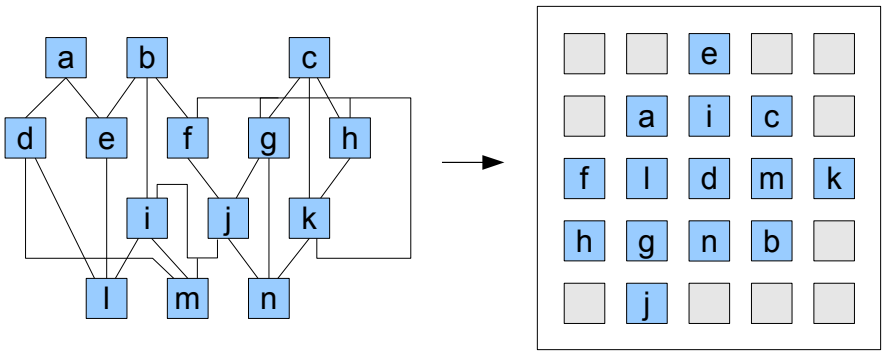


Figure 2.6: Illustration of Placement (from [16])

cluster. Greedy clustering algorithms are fast and area efficient, but due to a lack of backtracking, they can get trapped in local minima.

For the experiments performed in this thesis, a replica of the iRAC clustering algorithm was used. iRAC clustering has been shown to produce a low routed channel width solution and good delay performance when compared to other clustering approaches.

2.2.4 Placement

The placement step attempts to find an optimal arrangement for each of the CLBs in the array. Each CLB in the FPGA can be placed in fixed locations in the array. Placement tools will typically try to find a location for each CLB, such that a cost function is minimized. For example, timing-driven placement attempts to minimize the lengths of the critical and near-critical paths; bounding-box placement will attempt to reduce the sum of the half-perimeter bounding boxes for all nets in the circuit; CMap [32] attempts to reduce peak interconnect demand by balancing predicted interconnect demand across the device; iRAP [24] attempts to balance local interconnect demand by matching the Rent parameter

of the placement with the architectural Rent parameter; the Rent parameter is a measure of how tightly interconnected a circuit is.

Placement algorithms generally fall into two categories: simulated annealing and analytical placement. Simulated annealing is a flexible technique that can be applied to any optimization problem. Different optimizations can be implemented just by changing the cost function. For FPGA placement, the simulated annealing approach first starts with a random placement of CLBs. Pairs of CLBs then exchange locations. For each swap, a cost function evaluates the cost of the swap. If the cost decreases, the move is accepted. If the cost increases, the probability of accepting the swap depends on the current temperature. Initially, the temperature is set to accept all swaps, regardless of the cost. Gradually, the temperature decreases, reducing the probability that detrimental swaps are accepted. The acceptance of detrimental swaps allows the placement tool to possibly find a placement with globally minimal cost, instead of being trapped in local minima. However, due to the random nature of simulated annealing, it is a computationally intensive method to arrive at a placement solution. For this thesis, placement will be based on a simulated annealing approach.

Analytical placement uses systems of equations to solve the placement problem. This can potentially be much faster than simulated annealing. For example, in force-directed placement, CLBs are modeled as a system of particles being connected by a system of springs. The final location of each CLB is determined by solving the system such that the system is at a state of equilibrium. A drawback of analytical placement is that illegal placements may occur due to CLBs overlapping each other; each CLB can only be placed in a discrete location. Legalization methods must then be introduced to address this issue.

A drawback of the traditional CAD flow is that clustering, placement, and

routing occur in sequence. Changes to clustering are able to significantly affect circuit structure. Yet, accurate information from the placement step regarding wirelength, timing, and routability are not available during clustering. Some techniques combine clustering and placement to overcome this issue. For example, SCPlace [7] extends the simulated annealing based placement method to allow BLEs to be swapped between CLBs while optimizing for wirelength and timing. Work by [30] utilized node duplication and a novel depth-optimal initial clustering solution with combined clustering and placement to reduce critical path delay.

2.2.5 Routing

The routing step determines which wires on the FPGA device will connect the signals between CLBs. In general, routing algorithms are classified into single or two-step algorithms. The two-step approach first performs global routing, then detailed routing. In global routing, a signal is assigned to input-output pins and a routing channel. In a subsequent detailed routing step, the signal is assigned to a specific track in the routing channel. Examples of two-step routing approaches include [6], [17], and [25]. Single-step routers combine global and detailed routing into a single step. Examples include [19], [20], and [22].

The routing algorithm used in this thesis is the PathFinder [20] negotiated-congestion routing algorithm contained within VPR. The PathFinder algorithm initially maze routes all nets in the circuit. This will lead to some routing resources being overused due to sharing from multiple signals. Those shared resources are assigned a cost and all nets are ripped up and re-routed. The cost for overused resources iteratively accumulates until enough nets avoid the use of these resources such that only one net is routed on each wire, meaning the circuit can be routed.

2.3 Un/DoPack CAD Flow

The traditional CAD flow is not typically equipped to address hard channel width constraints of FPGA devices. Once the routing fails, designers need additional flexibility to produce a routable solution. Un/DoPack [29] is a fully automated congestion-aware CAD flow which performs re-clustering at a local level to meet a hard channel-width constraint.

Re-clustering of local regions is especially important in the case of circuits with large interconnect variation, such as system-on-chip circuits, which may consist of many different subcircuits, each having significantly different interconnect demands.

While clustering tools exist which maximize logic utilization by fully packing CLBs, various authors [12][27] have shown that the best performance may result from a balance between logic utilization and interconnect demand. Work by [4][26][29] showed that overall area could be reduced by packing CLBs to less than 100% capacity. Since FPGA area is dominated by interconnect, reducing the overall interconnect requirements by balancing local routing demand with logic utilization can produce a net decrease in FPGA area.

The clustering tool in [27] performs depopulation uniformly across the design, which depopulates uncongested regions along with congested regions.

Shown in Figure 2.7, Un/DoPack [29] reduces interconnect only in localized congested regions of the FPGA by iteratively reclustering regional BLEs into an increased number of CLBs. The user provides the following inputs to Un/DoPack: a circuit description, architecture description, target channel-width constraint, and an array size constraint. Iteratively, the MRCW of a circuit is reduced by spreading local areas of high congestion. This happens until the MRCW meets

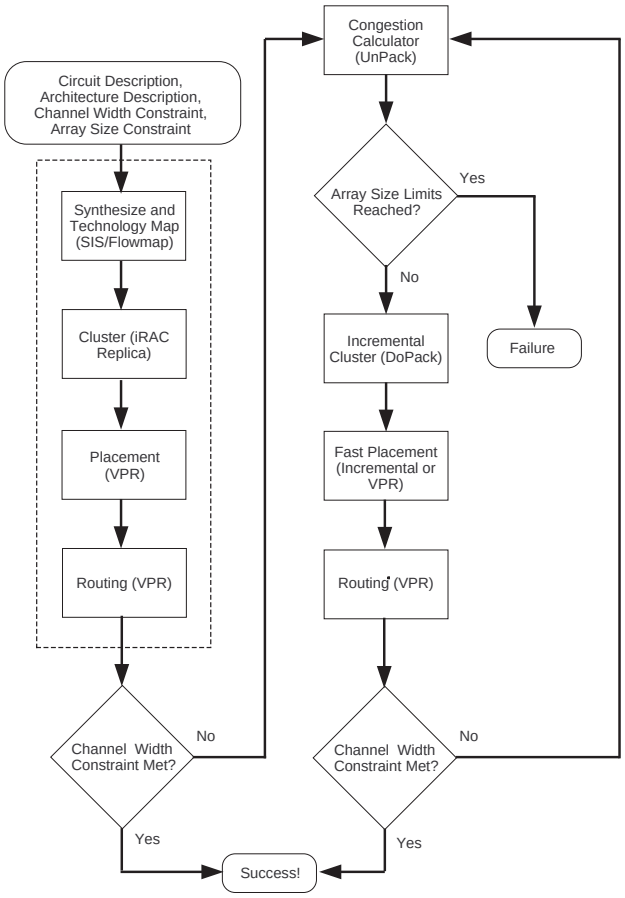


Figure 2.7: Un/DoPack CAD Flow (From [29])

the specified channel-width constraint.

Initially, a traditional CAD flow using SIS/FlowMap and VPR is run. This is shown inside the dashed outline in Figure 2.7. Any packing, placement, and routing algorithm can be used; we use iRAC and VPR. If the specified target channel-width constraint is met on the first pass, it is done. If the channel-width constraint cannot be met, the iterative portion of the Un/DoPack flow is invoked, which consists of the steps described below.

- The first step of the iterative portion, the UnPack step, determines which regions to depopulate. A region should be depopulated if the local interconnect demands of the region exceed the specified channel-width constraint.

Previous work presented two region selection schemes: single and multiple region schemes. These schemes select regions of a fixed radius, and calculate a new cluster size. The new cluster size is calculated such that enough whitespace is introduced to expand the number of CLBs in each region by a pre-determined amount.

- The second step repacks the BLEs in the selected regions. The clusters are packed less than 100% full using new cluster sizes determined by the UnPack step. BLEs from each region are individually reclustered with other BLEs from the same region. Any clustering algorithm can be used; for this work we used iRAC as our clustering algorithm.

- The final step is to perform placement and routing. If the final result is a routable solution, the CAD flow will exit. If the circuit remains unroutable after this step, the algorithm will iterate: the congestion information from routing will be used in the UnPack step to determine which regions should be depopulated next.

For placement, Un/DoPack uses a incremental placer, RePlace [18], which preserves the placement stability in each iteration. At each iteration, Un/DoPack is creating additional CLBs. These CLBs should be placed in close proximity to other CLBs from the same region. Existing CLBs are shifted to create room for the newly created CLBs. A low temperature anneal is then performed to optimize the placement. Since Un/DoPack is modifying only small localized regions of the FPGA, the incremental placement has the effect of significantly reducing runtime, when compared to a full VPR placement. Incremental placement also preserves the existing locations of CLBs in uncongested regions.

While Un/DoPack was shown to be very effective in reducing the channel-

width, runtime and area expansion can be further reduced through the use of better congestion-driven region selection and cluster-size calculation techniques.

In the following sections, we will discuss the various Un/DoPack schemes against which this work will compare.

2.3.1 Baseline Un/DoPack

The Baseline version of Un/DoPack depopulates a large, single region of the device.

In each iteration, a single region is selected by marking all the CLBs in a circular area, centered on the CLB with the highest congestion label, closest to the center of the array. The congestion label is the maximum of the number of signals in the x or y channel immediately adjacent to the CLB. In this work, we will use the original parameters for region size, from [29]. A circular region with a radius of array_size / 4 will be used.

The number of new CLBs created is equal to the number of CLBs in one row plus one column of the FPGA array. Equation 2.2 illustrates the increase in CLBs using the Baseline Un/DoPack approach. We will be comparing the area and runtime performance of each algorithm against the Baseline version of Un/DoPack.

$$num\_new\_CLBs\_baseline = \sqrt{(num\_CLBs\_in\_FPGA) * 2} + 1 \tag{2.1}$$

$$new\_region\_CLBs = \frac{num\_region\_LEs}{num\_CLBs\_in\_region + num\_new\_CLBs\_baseline} \tag{2.2}$$

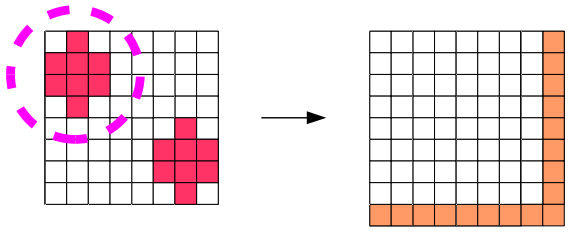


Figure 2.8: Region Selection and Whitespace Insertion - Baseline Un/DoPack

2.3.2 Fine-Grained Un/DoPack

As one of the observations in [28], authors noted that adding very small amounts of whitespace at each iteration produced superior area results compared to the Baseline approach, at the expense of increased runtime. This is referred to as the Fine-Grained approach. At each iteration, a single small region is selected for depopulation. We will be using the same region size as in [28], which is a circular region of radius array_size / 8.

The number of new CLBs in each region is determined by Equation (2.4). The Fine-Grained Un/DoPack method offers the least amount of area inflation, for the same channel width targets. Therefore, we will be comparing our work against the area performance of Fine-Grained Un/DoPack.

$$num\_new\_CLBs\_finegrained = \sqrt{(num\_CLBs\_in\_region) * 2} + 1 \tag{2.3}$$

$$new\_region\_CLBs = \frac{num\_region\_LEs}{num\_CLBs\_in\_region + num\_new\_CLBs\_finegrained} \tag{2.4}$$

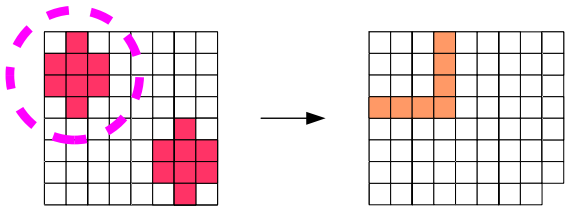


Figure 2.9: Region Selection and Whitespace Insertion - Fine-Grained Un/DoPack

2.3.3 Multiregion Un/DoPack

The multiple region approach presented in [29], referred to as Multiregion Un/DoPack, reduced runtime by depopulating multiple regions simultaneously in each iteration.

Regions are selected by centering the region on the CLB with the largest congestion value, closest to the center of the array. Once marked, CLBs cannot belong to any other region. Iteratively, all CLBs with a congestion value over the target channel width constraint are marked in this way, until no other unmarked CLBs have a congestion value over the target channel width constraint. We use the region size presented in [29], which is a circular region with a radius of array_size / 10.

In this approach, there is no restriction on how many CLBs are created in each iteration. Each region in the Multiregion scheme grows proportionally to the peak congestion in each region and an empirically determined scaling factor. In each iteration, the new number of CLBs in congested regions are calculated according to Equation 2.5 and Equation 2.6.

$$\alpha = 45 \cdot region\_radius \tag{2.5}$$

$$new\_region\_CLBs = \alpha \cdot \left(\frac{highest\_clb\_label\_of\_region}{channel\_width\_constraint} - 1 \right) \tag{2.6}$$

Of the previous Un/DoPack approaches, Multiregion Un/DoPack requires the lowest runtime. We will be comparing our work against the runtime peformance of Multiregion Un/DoPack.

Chapter 3

Multiple Region Depopulation with Congestion-Driven Metrics

The work presented in this thesis is based on the Un/DoPack CAD flow. We extend this approach to use congestion information presented by the placement and routing stages to more accurately determine the amount of whitespace to insert at each iteration. We introduce congestion-driven techniques that utilize local congestion metrics to reduce runtime and area inflation. Two new approaches are presented which explore the use of congestion information to determine which regions will be reclustered in each iteration of the Un/DoPack flow. First, we present an approach to help the CAD flow to better select congested regions. Second, we relax the constraint on how many CLBs are created in each iteration, and apply an interconnect demand model to determine the amount of whitespace to insert in each iteration. Finally, we introduce a congestion-aware placement algorithm to show that it is possible to further improve the overall quality of the Un/DoPack flow through the inclusion of other congestion-aware tools.

3.1 Budgeted Multiregion Un/DoPack (BMR)

Our first observation is that while previous work showed that depopulation is effective in reducing local interconnect congestion, it is important to correctly select congestion areas. A CAD flow was built which modified the region selection technique of Un/DoPack. We will refer to this as the Budgeted Multi-Region

(BMR) approach for the remainder of this work.

Multiple small regions are utilized to capture congested areas more accurately than a large single region. In addition, the number of CLBs created in each iteration is limited by a budget which is the same as in the Baseline Un/DoPack Flow; this budget is such that the number of CLBs in each iteration increase the FPGA array size by 1 row and 1 column. Each region is depopulated by a fixed amount equal to the Fine-Grained version of Un/DoPack. This has the effect of inflating area as much as the Baseline method, but the budget is spread to multiple regions.

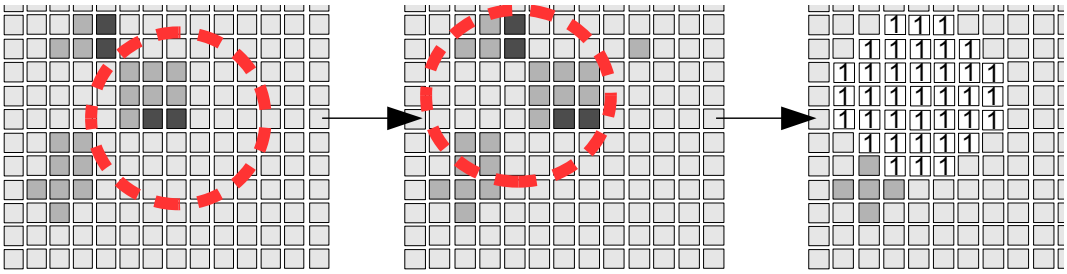
We will show that despite the overall same area growth in each iteration, we can converge more quickly to a routable solution.

3.1.1 Region Selection

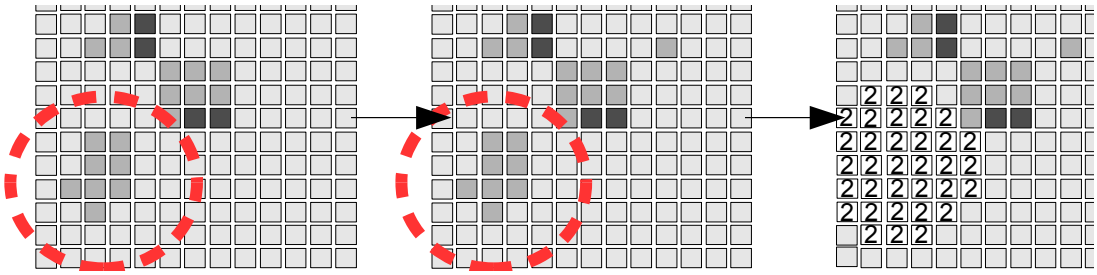
Instead of selecting a single large region, such as in Baseline Un/DoPack, our BMR approach creates regions that cover all congested CLBs. The details of each step in region selection is as follows:

- The first congestion region is selected by finding the CLB with the highest congestion label, closest to the center of the array. This forms an initial x,y center location for a circular windowed region of size bmr_radius which will be marked for depopulation. In this work, bmr_radius is array_size / 10. This step is shown in the leftmost illustration in Figure 3.1(a). The initial region is centered on the most-congested (darkest in the illustration) CLB, closest to the center of the array.

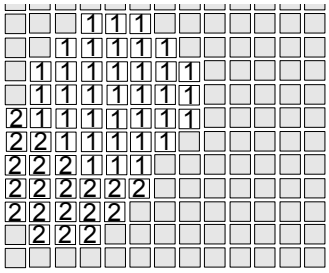
- The window center is adjusted slightly (up to $\pm\frac{bmr\_radius}{2}$) in each direction by using force-directed shifting. Force vectors between each CLB in the win-



(a) Selection of the First Region



(b) Selection of the First Region



(c) CLBs are Marked for Depopulation

Figure 3.1: Example of Region Selection

dow and the center of the region are calculated using the congestion values for each CLB. The sum of these vectors produces a direction in which to move the window. A binary search along this direction between the starting point and the farthest possible new starting point is then used to determine the window location that encompasses the largest average congestion. The furthest that the window can shift is a distance of up to $\pm\frac{bmr\_radius}{2}$ away from the original window center. This produces a force-directed move to shift the region to encompass the most amount of congestion. The center illustration in Figure 3.1(a) shows that location of the region has shifted.

The congested CLBs near the top left of the array create a net force which cause the region to shift.

- The CLBs in this selected region are marked as belonging to this region. This is shown in the rightmost illustration in Figure 3.1(a)

- The next congestion region is selected by finding the next CLB with the largest congestion value, closest to the center of the array. The center CLB is only chosen from CLBs not already selected. This is shown in the leftmost illustration in Figure 3.1(b).

- Force-directed shifting is applied to the new region. The region center is not allowed to shift into an already selected region. However, once the region location is determined, CLBs overlapping with other regions are allowed to belong to this new region. This is shown in the center illustration in Figure 3.1(b).

- This continues until all CLBs exceeding the target channel-width constraint are covered by a region.

The force-directed move ensures the depopulation window region is repositioned so the CLB label peak value is still captured, but it will also capture as many other CLBs as possible that need depopulation. A list of all such regions is then sorted such that the regions with the highest average congestion are depopulated first. This ensures that the overall budget will be spent on the most-congested regions first. At each depopulation step, we determine the number of CLBs which will be added. Once a region is depopulated, subsequent regions will not be able to depopulate the CLBs that are already depopulated in this iteration. The regions are depopulated in sorted order until all congested regions are depop-

ulated. This is shown in Figure 3.1(c). Although region 1 and region 2 overlap, region 1 will be depopulated first because it is more congested. Subsequently, region 2 is depopulated but does not overlap with region 1.

In addition, those adjacent regions with the same target cluster size are merged together into a single combined "super-region", which may allow for better flexibility during reclustering.

3.1.2 Whitespace Insertion

Whitespace insertion is achieved by reducing the cluster size of the CLBs, such that some BLEs become unused. For the BMR flow, we limit the growth in each region to an amount determined by Equation 3.1. However, the total number of new CLBs produced in each iteration is limited by a budget, defined in Equation 3.2.

$$num\_new\_CLBs\_region = \sqrt{(num\_CLBs\_in\_region) * 2} + 1 \tag{3.1}$$

$$new\_CLBs\_bmr = \sqrt{(num\_CLBs\_in\_FPGA) * 2} + 1 \tag{3.2}$$

For each region marked for depopulation, we subtract the number of new CLBs created in each iteration against the budget until the budget is exhausted. In some cases, the target cluster size can not be attained because the remaining budget is insufficient. In this case, all of the remaining budget will be applied to that region and the cluster size will be calculated accordingly.

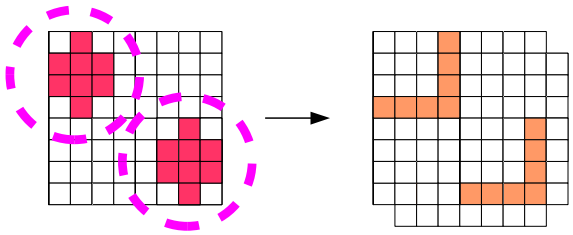


Figure 3.2: Region Selection and Whitespace Insertion - BMR Un/DoPack

3.2 Congestion-Model Multiregion Un/DoPack (CMR)

In addition to improved region selection, we experimented with using a channel-width demand model to improve the accuracy of the whitespace insertion. We will term this scheme the Congestion-Model Multiregion Un/DoPack (CMR). We utilize the same region selection method as in the above BMR flow, but we extend the whitespace insertion method to utilize congestion information to determine a target cluster size. The overall budget is removed to allow as much depopulation in each iteration to occur as needed.

Of the interconnect models available in previous work, the most applicable to this work consist of those models which predict wire length and channel-width demand [13][14]. The work in [13] extends previous models to consider the routing inflexibility inherent in FPGAs. The advantage of this model is that we can use it to predict outcomes in interconnect demand based on decisions made during clustering. Since this model assumes its application to an entire FPGA, we make some simple assumptions to apply it to our local depopulation regions.

3.2.1 Modeling Regional Interconnect Demand

While most channel-width demand models predict the interconnect demand at a global level, we are interested in determining, on a region-by-region basis, how much whitespace insertion is necessary for each congested region to become routable. Thus we create the following simple model of local interconnect demand to allow the application of a global model, shown in Equation 3.3.

$$total\_region\_demand = region\_internal\_demand + region\_external\_demand \tag{3.3}$$

Interconnect demand for a region of CLBs can be generally categorized in two types: internal interconnect, which is the interconnect needed to route between CLBs inside a region, and external interconnect, which consists of routing that connects CLBs outside the region, but pass through the region without connecting to any CLBs inside the region.

While depopulation directly affects the internal interconnect demand through whitespace insertion, we cannot directly reduce interconnect demand from external routing by whitespace insertion into a region. Instead, we must separately account for its effects by identifying the contribution to the channel-width demand inside a region caused by these external nets, as shown in Equation 3.3. For each region, our goal is to reduce total_region_demand to meet our channel-width constraint by reducing region_internal_demand.

We apply a congestion-estimation model, Wirelength-per-Area [31], to the internal and external nets separately. We can compare the amount of average interconnect demand from internal nets to the average interconnect demand of

external nets. Combining this with post-routing information, we can then estimate the actual interconnect demand due to external and internal nets.

For simplicity, we assume that subsequent iterations will produce relatively the same amount of external-net congestion in the next iteration. Although this is a simplification, interconnect demand from external nets are typically less than interconnect from internal nets. This is intuitive since local routing should mostly originate from logic inside regions. We leave the influence of inter-region effects on congestion to future work. The channel-width estimation model using Equation 3.5 from [13], is then used to determine the amount of whitespace to insert for the next reclustering step.

3.2.2 Modeling Internal Demand

The interconnect model presented in [13] is shown in Equation 3.5. This model is an extension of El Gamal's master slice interconnect model [14], which predicts the channel-width for a fully flexible FPGA. This is shown in Equation 3.4. W_{abs_min} is the channel-width required for a fully flexible FPGA, while λ is the average number of used inputs and $\overline{R}$ is the average point-to-point wirelength. The channel-width determined from the master slice interconnect model does not account for additional channel-width required due to routing inflexibility caused by wire segment length, switch blocks, and connection blocks. Instead, [13] accounts for these by including additional terms, shown in Equation 3.5.

$$W_{abs\_min} = p\frac{\lambda\bar{R}}{2} \tag{3.4}$$

$$
\begin{aligned}
W &= W_{abs\_min} \\
&+ \frac{1}{\beta}\left(\frac{W_{abs\_min}}{F_s}\right)\left(\frac{W_{abs\_min}}{F_{Cin}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{F_{Cout}}\right)^{\alpha_{out}} \\
&+ \frac{\lambda(L-1)}{4}\left(1+\frac{1}{F_{Cin}^{\alpha in}}\right)
\end{aligned}
\tag{3.5}
$$

In Equation 3.5, W is the estimated peak channel-width. Post-placement, we can measure the average number of used inputs per CLB, λ. F_s is switch block flexibility, F_{Cin} and F_{Cout} denote connection block flexibilities for inputs and outputs, and L is wire segment length; these values can be determined from the FPGA architecture. We use the values 1.4, 0.5, and 0.25 for β, α_{in}, and α_{out}, respectively, as presented in [13]. We assume, as in [13], that $\overline{R}$ remains constant with cluster size. Therefore, by using the measured peak channel-width for a region and average number of used inputs, we can solve Equation 3.5 for $p \cdot \overline{R}$. With these constants set, we substitute W for the target channel-width and solve Equation 3.5 for W_{abs_min}. From Equation 3.4, we can then solve for λ. Therefore, by re-clustering the region at the next iteration to meet the λ constraint on the number of used inputs, this region should become routable.

In our implementation, the clustering tool iteratively reclusters a region with a progressively lower cluster-size until the average number of used inputs constraint is met. Since the clustering runtime is very small, the increase in overall runtime is negligible. In addition, we retain the flexibility of the clustering tool to use different clustering methods.

3.3 Congestion-Aware Placement

A second goal of this work is to show that a congestion-aware placement tool improves the overall quality of the Un/DoPack flow, but is insufficient as a

replacement to Un/DoPack. While several congestion-aware placement tools exist [32][24][5], work presented in [32] follows a philosophy complimentary to Un/DoPack by optimizing placement to reduce local congestion.

$$Cost = coeff * \sum_{i=1}^{N_{nets}} q[i](bb_x(i) + bb_y(i)) \tag{3.6}$$

The placement approach in [32], referred to as Bounding Box Overlap placement, uses a congestion estimation map to create a coefficient. This coefficient is multiplied with the bounding box cost function in the VPR placement tool to penalize swaps which lead to congested placements. The modified cost function is shown in Equation 3.6. For each net i, the horizontal and vertical spans, $bb_x(i)$ and $bb_y(i)$, are added and multiplied with the q(i) factor which compensates for the fanout of the net. This is summed over all nets and multiplied with the coefficient calculated from the congestion map. The coefficient is calculated using Equation 3.7.

$$coeff = \left(\frac{\Sigma_{i,j}U_{i,j}^2}{nx \cdot ny} / \left(\frac{\Sigma_{i,j}U_{i,j}^2}{nx \cdot ny}\right)^2\right) \tag{3.7}$$

In Equation 3.7, $U_{i,j}$ is a CLB label in the congestion estimation map used. The congestion estimation map in [32] uses an approach referred to as Bounding Box Overlap. The congestion estimation map indicates how many bounding boxes overlap each CLB. Since we are able to use any method to generate a congestion estimation map, we also took the opportunity to explore a slight variation to Bounding Box Overlap heuristic. In [31], authors explained that a related

Circuit	VPR Default (tracks)	VPR BB (tracks)	BB Overlap (tracks)	Wirelength per Area (tracks)
stdev0	96	95	92	93
stdev002	96	93	94	86
stdev004	101	97	98	92
stdev006	89	89	90	86
stdev008	119	116	115	106
stdev010	153	150	152	139
stdev012	145	145	141	138

Table 3.1: Maximum MRCW Comparison of Placement Schemes

heuristic, Wirelength per Area, produces a congestion map which better indicates relative local amounts of interconnect demand when compared to Bounding Box Overlap. We created congestion maps using the Wirelength per Area method and applied this to the congestion-aware placement tool.

Table 3.1 illustrates the effect of congestion-aware placement on the maximum MRCW of our benchmark suite, and Table 3.2 compares the runtime performance for each placement scheme. While much slower in runtime, results show that a congestion-aware placer consistently reduces the number of routing tracks. The runtime and maximum MRCW results obtained using VPR default, VPR bounding box, Bounding Box Overlap, and Wirelength per Area placement approaches, are compared in Table 3.1 and Table 3.2. We note that the Wirelength per Area method indeed performs slightly better in reducing the maximum MRCW, although both produce consistently good results. Our experiments combine our Congestion-Model Multiregion version of Un/DoPack with the congestion-aware placement, using the Wirelength per Area method. This will be called CMR-CAP. The congestion-aware placement is integrated with the incremental placement tool, RePlace.

Circuit	VPR Default (seconds)	VPR BB (seconds)	BB Overlap (seconds)	Wirelength per Area (seconds)
stdev0	2406	783	10063	10918
stdev002	2207	882	10226	10639
stdev004	2170	831	8792	9694
stdev006	1704	692	8794	9803
stdev008	1810	776	9602	10204
stdev010	2279	1037	10394	11942
stdev012	1875	1063	12126	13808

Table 3.2: Runtime Comparison of Placement Schemes

Chapter 4

Results

We compared the runtime and area performance of Baseline Un/DoPack to the following schemes: Multiregion Un/DoPack, Fine-Grained Un/DoPack, Budgeted Multiregion Un/DoPack (BMR) and the Congestion-Model Multiregion Un/DoPack (CMR). We also performed experiments which combined the Congestion-Model Multiregion Un/DoPack approach with a congestion-aware placement tool (CMR-CAP).

4.1 Experimental Methodology

This section discusses the experimental framework to evaluate our algorithmic improvements.

The results in this work are normalized to the baseline Un/DoPack flow presented in [29]. Baseline Un/DoPack has the following characteristics:

- Congestion calculator with single region depopulation
- Clustering algorithm which is a replica of iRAC
- RePlace, incremental placer presented in [18], using default VPR placement (timing and wirelength driven placement)
- VPR flags: pres_fac_mult 1.3, max_router_iterations 100
- FPGA architecture with LUT size k = 6, cluster-size N = 16, inputs per cluster I = 51, and a wire length of L = 4

The experiments were conducted on a single core of a Xeon X5355 2.66 GHz processor with 16GB of RAM. The maximum MRCW of each circuit, was determined from VPR with the binary search option set. The verify_binary_search option in VPR was used to ensure that the lowest routable channel width was measured. All versions of Un/DoPack were run on our servers including the following Un/DoPack schemes: Baseline, Fine-Grained, BMR, and CMR. All VPR simulations used an overuse penalty factor growth factor, pres_fac_mult, of 1.3 and the maximum number of router iterations, max_router_iterations, set at 100. The channel-width constraints are the same as those presented in [29] using the benchmark circuits described below.

We used the benchmark circuits from [29]. Each of the benchmark circuits have the following characteristics: 40013 LUTs, 241 inputs, 120 outputs, and approximately 52000 nets. These circuits are designed to help examine the performance of CAD tools for large, system-on-chip designs which are composed of subcircuits with varying amounts of interconnect demand. The benchmark circuits created in [29] were generated by mimicking the properties of MCNC benchmark circuits [21] as subcircuits in one large, synthetic circuit using the generator GNL. GNL allows the user to specify the overall Rent parameter of the circuit, and also the Rent parameter of individual subcircuits. The average Rent parameter is 0.65 for each benchmark circuit, but the standard deviation of the Rent exponent for the subcircuits is varied. The result is a set of benchmark circuits which contain some circuits with uniform local interconnect demand across the circuit, while others have regions of high local interconnect demand.

For each of the Un/DoPack multiregion depopulation methods, we use a depopulation radius of array_size / 10. This is the same region size of Multiregion Un/DoPack presented in [29]. We use a depopulation radius of array_size / 4 for

Baseline Un/DoPack, and a depopulation radius of array_size /8 for Fine-Grained Un/DoPack, in accordance to [28]. Multiregion Un/DoPack will be our target approach for a low-runtime version of Un/DoPack, while Fine-Grained Un/DoPack will form the target for the low-area version of Un/DoPack.

A separate set of experiments were performed to examine the effect of combining a congestion-aware placement tool with the Congestion-Model Un/DoPack approach. All parameters for this set of experiments are identical to those mentioned above, with the exception of the inclusion of the congestion-aware placement cost function within the incremental placer.

4.2 Previous Un/DoPack Schemes

Figure 4.1 shows the area versus runtime results for Multiregion and Fine-Grained Un/DoPack for each of the benchmark circuits used in this thesis, over a range of channel width constraints. The horizontal axis indicates the runtime for each benchmark, normalized to the runtime of Baseline Un/DoPack. The vertical axis indicates the area for each benchmark, normalized to the area of Baseline Un/DoPack. In this work, area is considered the sum of routing area and CLB area. Area is calculated in the same way as VPR [3], where the layout area of an individual transistor is expressed in units of minimum-width transistor areas. The vertical axis indicates the runtime of each scheme, normalized to the runtime of Baseline Un/DoPack.

Our results show that of the previous Un/DoPack schemes, the Fine-Grained approach produces the best results in terms of area; the area is reduced by up to 30% when compared to Baseline Un/DoPack. However, this has a large runtime penalty; due to the small number of CLBs inserted at every iteration, many

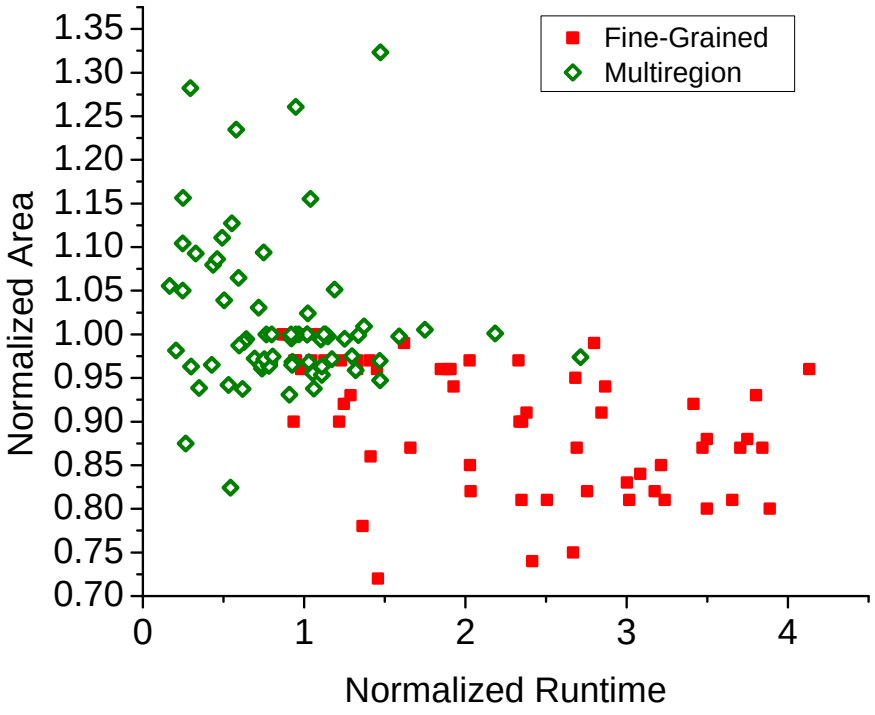


Figure 4.1: Area versus Runtime - Fine-Grained and Multiregion Schemes, Normalized to Baseline

iterations are needed, especially for lower target channel-width constraints. The runtime increases accordingly. Multiregion Un/DoPack reduces runtime by depopulating multiple regions simultaneously, while inserting whitespace proportional to the peak congestion value of a region. As expected, Multiregion Un/DoPack outperforms the runtime of Baseline Un/DoPack, improving runtime by up to 6x. Area performance also improves in general by up to 17.5% over Baseline Un/DoPack. However, unlike Fine-Grained Un/DoPack, which consistently reduces area up to 28%, Multiregion Un/DoPack also produces worse area results, inflating area up to 32% more than Baseline Un/DoPack.

4.3 Budgeted Multiregion Un/DoPack

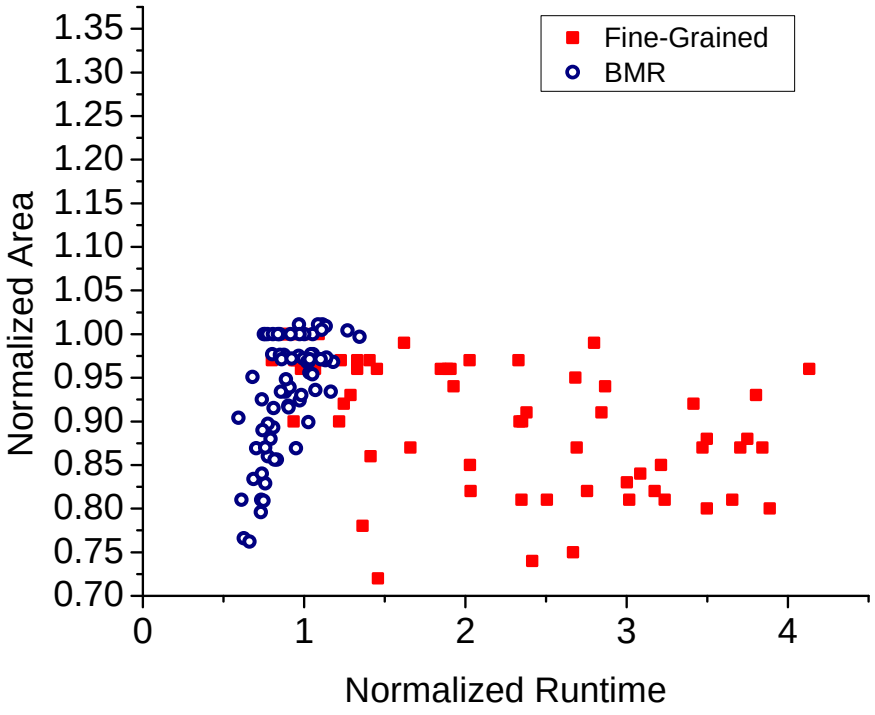


Figure 4.2: Area versus Runtime - Fine-Grained and BMR

Figure 4.2 shows the area versus runtime results for BMR and Fine-Grained Un/DoPack for each of the benchmark circuits used in this thesis, over the same range of channel width constraints. We observe that BMR Un/DoPack is able to maintain area performance which is comparable to Fine-Grained Un/DoPack, and yet runtime is significantly faster. The general trend for Fine-Grained Un/DoPack is that as the area performance improves against Baseline Un/DoPack, but runtime also increases. In BMR however, the opposite trend is apparent; a decrease in area is accompanied by a decrease in runtime; this is despite BMR creating as many CLBs as Baseline Un/DoPack in each iteration. The speedup can be explained, in some cases, by a reduction in iterations needed to reach a channel

width constraint. Depopulating multiple congested regions simultaneously allows BMR to converge faster by depopulating multiple locations.

The area performance of BMR is better than Baseline Un/DoPack. A smaller CLB increase in each region may prevent regions from being over-depopulated. Accurate region selection reduces the likelihood that uncongested CLBs are depopulated, due to non-circular congestion regions and force-directed shifting. Compared to Baseline Un/DoPack, area is reduced by up to 23%, with the maximum increase over Baseline at 1.1%. This leads to the conclusion that increasing the total CLB budget for depopulation in each iteration is an important factor in reducing runtime. For the congestion in each region to be resolved as soon as possible, we require an adequate amount of depopulation. However, by reducing the amount of depopulation in each region to the minimum necessary amount, unnecessary area inflation can be reduced.

4.4 Congestion-Model Multiregion Un/DoPack

For the CMR scheme, the limit on area growth for each iteration is removed. Instead, a congestion model for whitespace insertion is added to determine how much whitespace to insert in each iteration.

Figure 4.3 shows a comparison of area and runtime performance between the Multiregion Un/DoPack and CMR Un/DoPack. Because of the removal of the budget, the number of iterations decreases dramatically compared to Baseline and Fine-Grained Un/DoPack. The reduction of iterations produces a runtime improvement that is comparable to Multiregion Un/DoPack. In our experiments, CMR Un/DoPack is able to achieve a 5.5x speedup over Baseline Un/DoPack, while Multiregion Un/DoPack is able to achieve a runtime improvement of 6x.

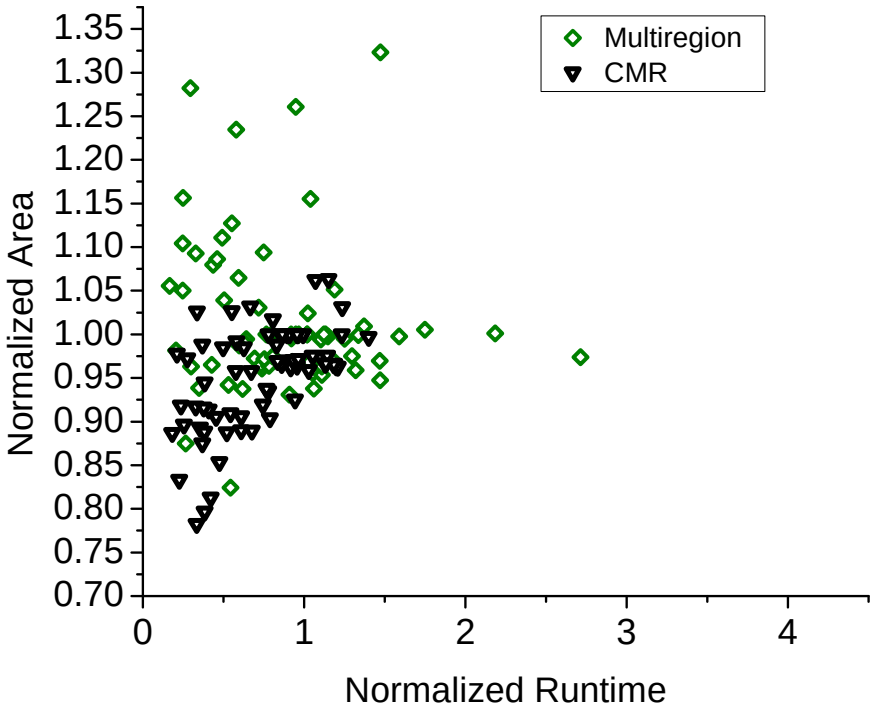


Figure 4.3: Area versus Runtime - Multiregion and CMR

However, we note that the area performance of CMR Un/DoPack is consistently better than Multiregion Un/DoPack. In some instances where Multiregion Un/DoPack has significant runtime speedup, the area growth is quite high. In contrast, an improvement in runtime is also accompanied by an area improvement for CMR in general.

Figure 4.4 shows a comparison of all multiple region depopulation methods. Both CMR and BMR show consistently better area performance than Multiregion Un/DoPack.

Figures 4.5 and 4.6 show examples of typical results for individual benchmarks. Shown is the area and runtime of Baseline Un/DoPack compared to other methods. The horizontal axis is minimum routable channel width, normalized to

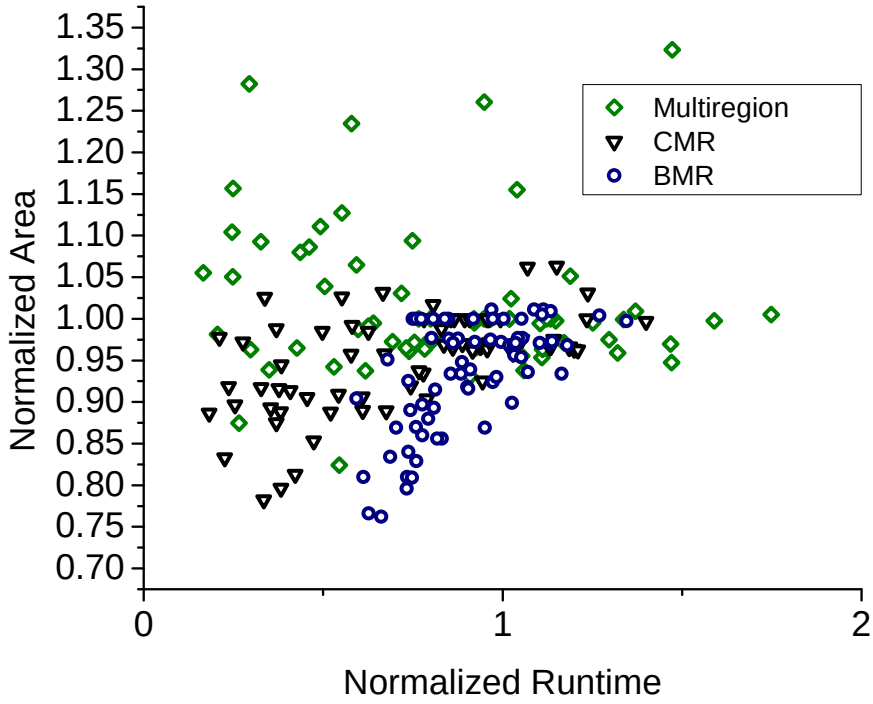


Figure 4.4: Area versus Runtime - BMR, Multiregion and CMR

the maximum MRCW. The maximum MRCW was determined by performing the traditional VPR CAD flow with the binary search option enabled for the routing step. This determines the minimum number of tracks needed to route each benchmark circuit before depopulation using Un/DoPack. Area is measured as the total transistor area of the logic and routing of the CLBs used. This captures the effect of successfully reducing the channel width as well as the effect of using more CLBs after depopulating. Runtime figures presented are normalized to the maximum MRCW for the circuit. A value of 0.6 on the x-axis means that the final routed channel width is reduced by 40%.

As expected, Fine-Grained Un/DoPack produces the best area with the worst runtime. Our CMR approach maintains or improves the runtime performance of

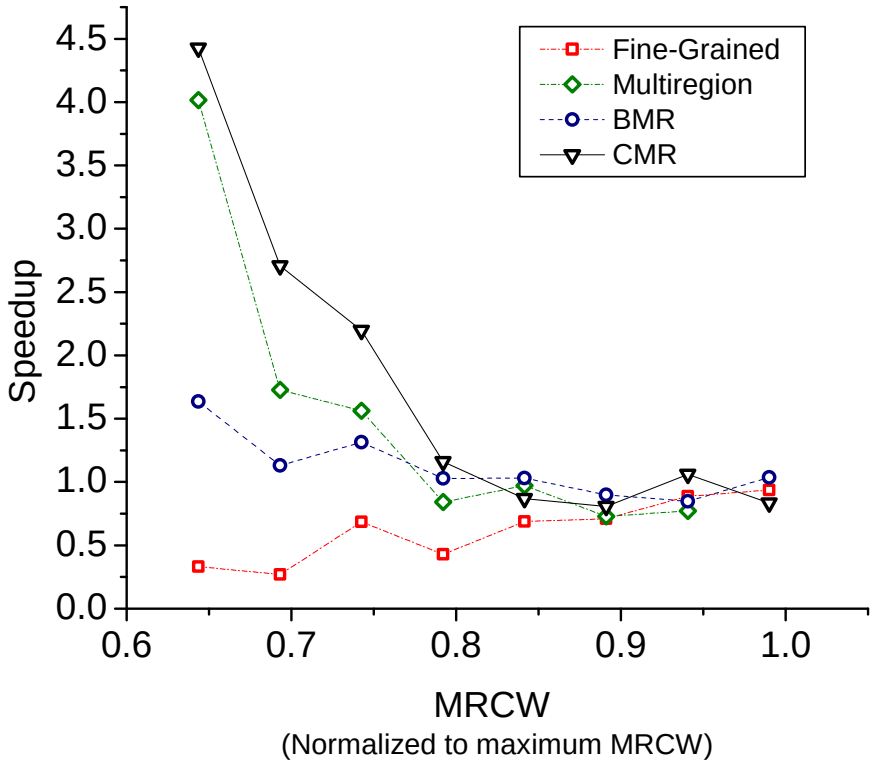


Figure 4.5: Runtime Comparison with Baseline Un/DoPack - Circuit Stdev004

the previous Multiregion approach, but also improves area.

Results for other circuits are similar to circuit Stdev004. However, in situations where the maximum MRCW is quite large, the performance of all approaches will be similar. This is because only a small number of regions need to be depopulated for the circuit to become routable. Therefore at larger channel width constraints, the number of iterations required for each approach is similar, and so the runtime is also similar. As the maximum MRCW is lowered, Baseline and Fine-Grained Un/DoPack perform less depopulation at each iteration than the Multiregion or Congestion-Model Multiregion approaches. This allows our approach to have a significant speedup at lower MRCW settings. In addition, our CMR approach

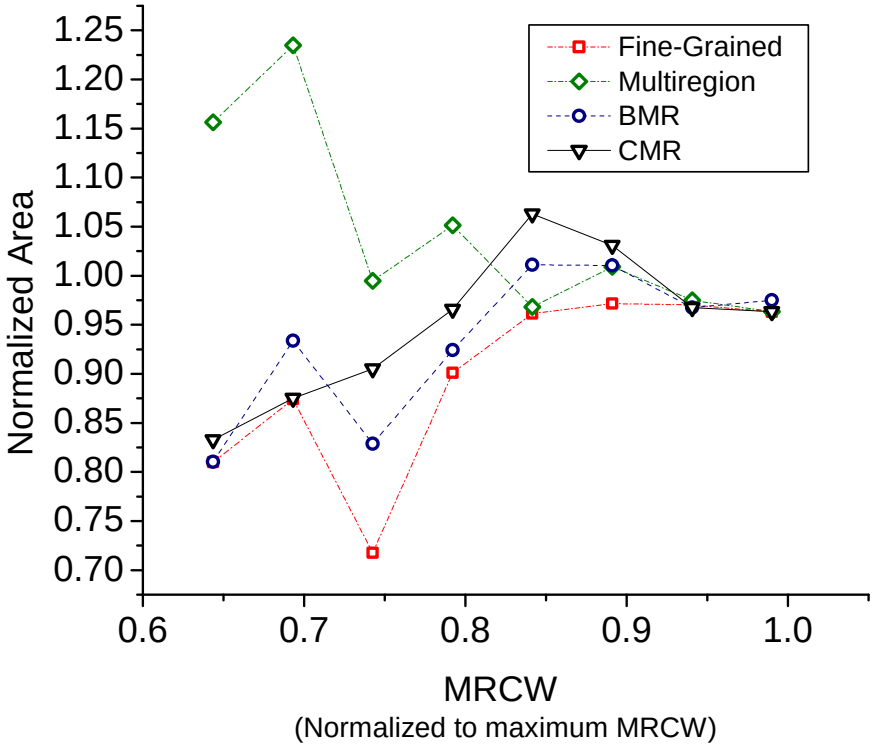


Figure 4.6: Area Comparison with Baseline Un/DoPack - Circuit Stdev004

reduces the over-depopulation of regions and area performance is improved even at low target channel widths.

4.5 Critical-Path Comparison

The critical path for each method are relatively similar. Typical critical path results, compared to Baseline Un/DoPack, are shown in Figures 4.7 and 4.8.

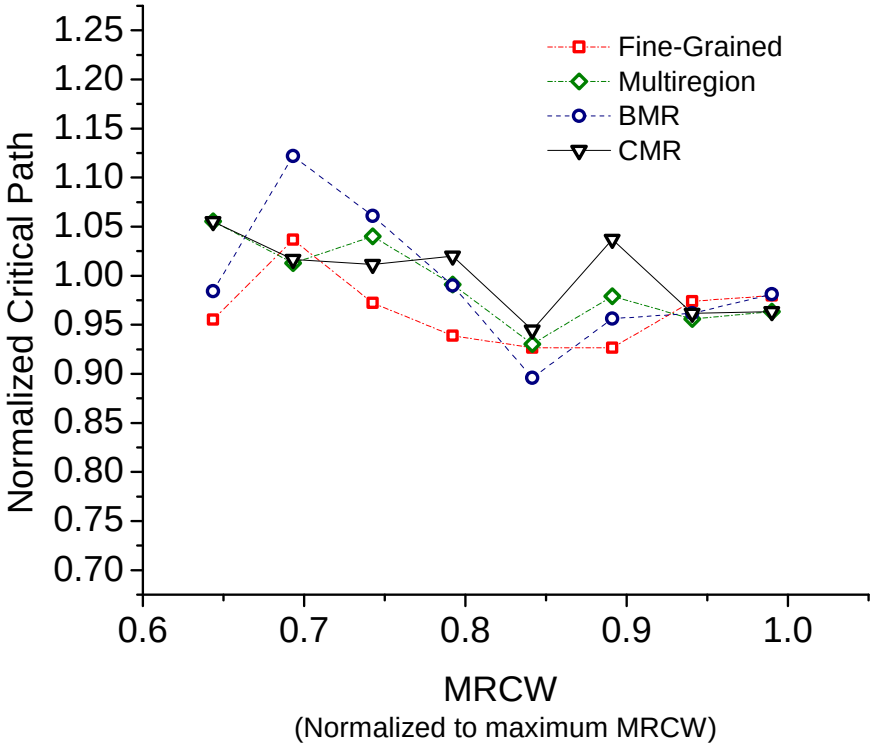


Figure 4.7: Critical Path Comparison - Circuit Stdev004

4.6 CMR with Congestion-Aware Placer

The inclusion of a congestion-driven placement tool can further improve the Un/DoPack approach. Figures 4.9 to 4.10 illustrate that combining CMR Un/DoPack with a congestion-aware placer consistently improves area.

As indicated in Table 3.1, this congestion-aware placement tool cannot by itself reduce the channel-width more than is possible with whitespace insertion. However, when combined with CMR, we can further limit area inflation of the Un/DoPack flow. This improvement can be significant, up to 25% better than CMR alone, as shown in Figure 4.10.Although this particular congestion-aware placement tool causes CMR-CAP Un/DoPack to be slower (4x slower than Base-

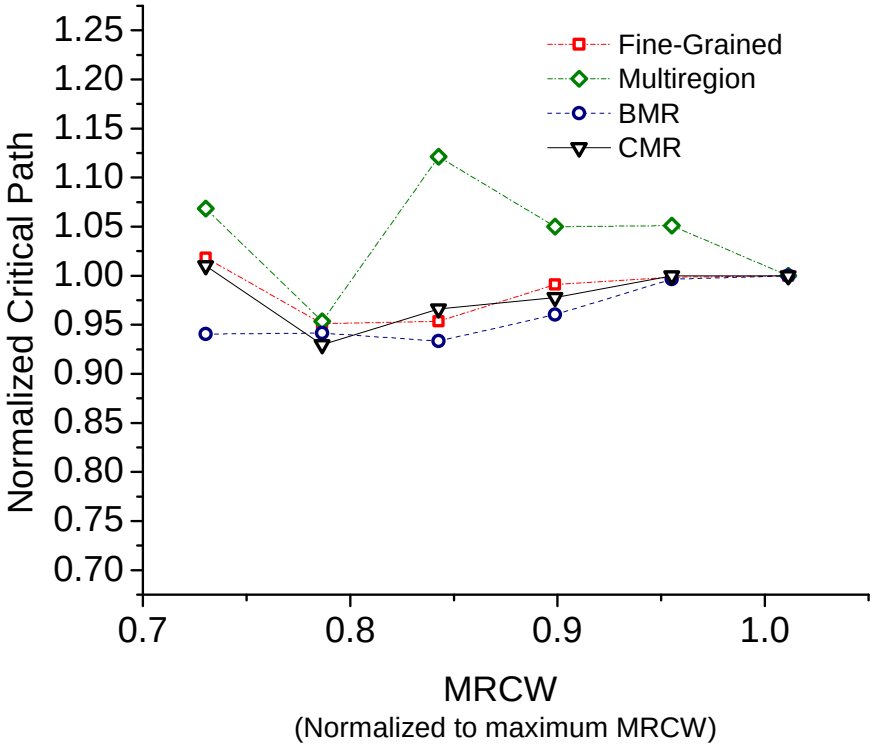


Figure 4.8: Critical Path Comparison - Circuit Stdev006

line Un/DoPack), our results indicate that the inclusion of other congestion-driven placement techniques can possibly reduce the area of the Un/DoPack flow. This illustrates that the inclusion of local congestion-driven techniques can further improve, but not replace, the Un/DoPack approach.

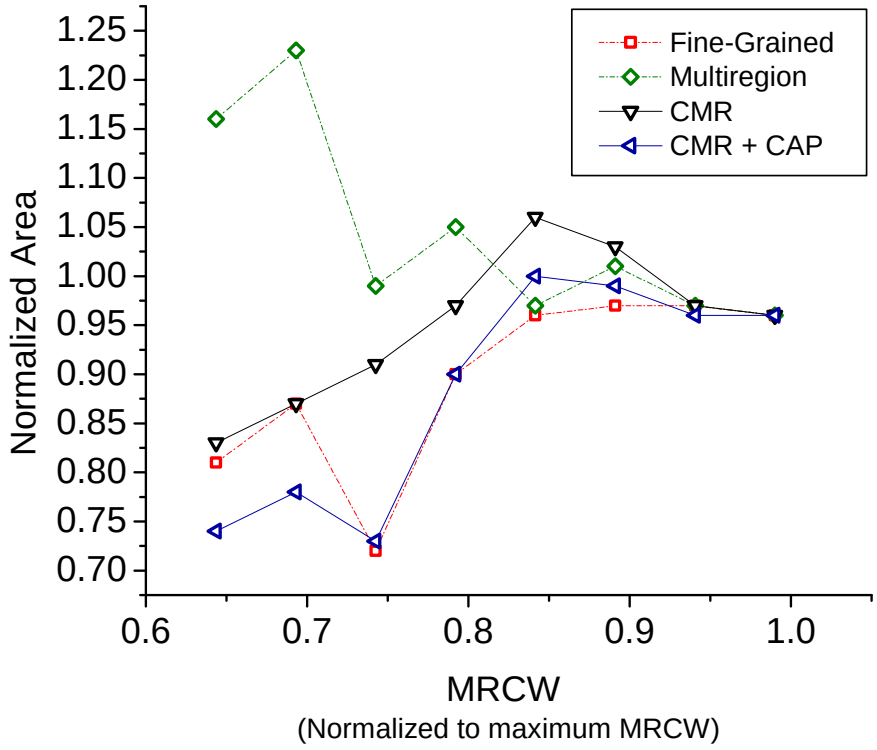


Figure 4.9: CMR with Congestion-Aware Placement - Stdev004

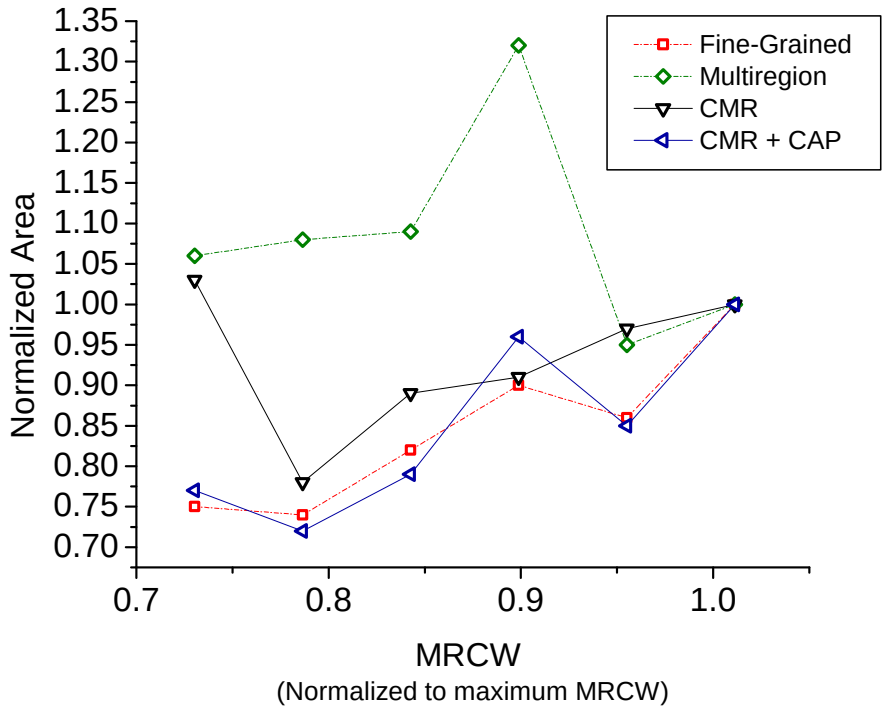


Figure 4.10: CMR with Congestion-Aware Placement - Stdev006

Chapter 5

Conclusions

In this thesis we show that we are able to effectively improve the performance of the Un/DoPack flow in area and runtime by effectively utilizing congestion information. This work presented methods to better select congested regions on an FPGA and calculate the amount of depopulation required at each iteration. Our Congestion-Model Multiregion approach is shown to improve runtime by a factor of up to 5.5 times and reduce area by up to 20%, compared to Baseline Un/DoPack. This allowed us to reduce channel-width up to 55%. We also showed that this CAD flow is complementary to using a congestion-aware placement method; the inclusion of a congestion-aware placement tool had positive effects on the overall area performance of Un/DoPack.

5.1 Future Work

5.1.1 Influence from Neighbouring Regions

To further improve the accuracy of congestion-driven whitespace insertion technique for each region, future work should consider the influence of depopulating neighbouring regions on the congestion. This is especially an important consideration when neighbouring regions are depopulated with different cluster sizes. The congestion model in this work assumes that the CLBs in each region will eventually be placed adjacent to CLBs from the same region. However, in reality,

it is possible that CLBs from one region to be placed into adjacent regions during the placement step.

5.1.2 Congestion-Driven Placement and Clustering

This work has shown that additional congestion-aware methods can further improve the area performance of Un/DoPack. Through fine-grained logic placement, SCPlace has been shown to produce more routable solutions at a better runtime than VPR simulated annealing. While SCPlace performs fine-grained placement, it does not introduce whitespace insertion into to help a non-routable solution become routable. The combination of a simultaneous clustering and placement method, combined with incremental placement and Un/DoPack whitespace insertion would further improve runtime and area performance.

Additionally, a large part of the runtime cost for each iteration is due to the time required for routing. We attempted to extract local interconnect demand information, using an estimator based on the Area-Per-Wirelength model [31]. While it is able to locate congested regions somewhat adequately, it does not provide the very accurate congestion information needed for our congestion-driven whitespace insertion. Additional research into estimation methods which estimate local congestion and routability information will help reduce the need to rely on full routing to determine accurate congestion information.

Bibliography

[1] Elias Ahmed and Jonathan Rose. The effect of LUT and cluster size on deep-submicron FPGA performance and density. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pages 3–12, Monterey, California, United States, 2000. ISBN 1-58113-193-3. URL `http://portal.acm.org/citation.cfm?id=329166.329171`.

[2] V. Betz and J. Rose. Cluster-based logic blocks for FPGAs: area-efficiency vs. input sharing and size. In *Custom Integrated Circuits Conference*, pages 551–554, 1997. doi: {10.1109/CICC.1997.606687}. URL `http://www.eecg.toronto.edu/~vaughn/papers/cicc97.pdf`.

[3] Vaughn Betz, Jonathan Rose, and Alexander Marquardt, editors. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999. ISBN 0792384601. URL `http://portal.acm.org/citation.cfm?id=553523`.

[4] Elaheh Bozorgzadeh, Seda Ogrenci-Memik, and Majid Sarrafzadeh. RPack: routability-driven packing for cluster-based FPGAs. In *Asia and South Pacific Design Automation Conference*, pages 629–634, Yokohama, Japan, 2001. ACM. ISBN 0-7803-6634-4. doi: 10.1145/370155.370567. URL `http://portal.acm.org/citation.cfm?id=370567`.

[5] Ulrich Brenner and Andre Rohe. An effective congestion driven placement

framework. In *International Symposium on Physical Design*, pages 6–11, San Diego, CA, USA, 2002. ACM. ISBN 1-58113-460-6. doi: 10.1145/505388.505391. URL http://portal.acm.org/citation.cfm?id=505391.

[6] Yao-Wen Chang, Shashidhar Thakur, Kai Zhu, and D. F. Wong. A new global routing algorithm for FPGAs. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 356–361, San Jose, California, United States, 1994. IEEE Computer Society Press. ISBN 0-89791-690-5. URL http://portal.acm.org/citation.cfm?id=191489.

[7] Gang Chen and Jason Cong. Simultaneous timing driven clustering and placement for FPGAs. In *Field Programmable Logic and Application*, pages 158–167. Springer Berlin / Heidelberg, 2004. URL http://www.springerlink.com/content/gyahkply7ltv78eu.

[8] J. Cong and Yuzheng Ding. FlowMap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(1):1–12, 1994. ISSN 0278-0070. doi: 10.1109/43.273754. URL http://citeseerx.ksu.edu.sa/viewdoc/summary?doi=10.1.1.22.9473.

[9] Jason Cong and Yuzheng Ding. On area/depth trade-off in LUT-based FPGA technology mapping. In *International Design Automation Conference*, pages 213–218, Dallas, Texas, United States, 1993. ACM. ISBN 0-89791-577-1. doi: 10.1145/157485.164675. URL http://portal.acm.org/citation.cfm?id=164675.

[10] Jason Cong and Yean-Yow Hwang. Simultaneous depth and area minimization in LUT-based FPGA mapping. In *ACM International Sympo-*

sium on Field-Programmable Gate Arrays, pages 68–74, Monterey, California, United States, 1995. ACM. ISBN 0-89791-743-X. doi: 10.1145/201310.201322. URL http://portal.acm.org/citation.cfm?id=201310.201322&coll=GUIDE&dl=GUIDE&CFID=47964849&CFTOKEN=55429101.

[11] Jason Cong and Yean-Yow Hwang. Structural gate decomposition for depth-optimal technology mapping in LUT-based FPGA designs. *ACM Transactions on Design Automation of Electronic Systems*, 5(2):193–225, 2000. doi: 10.1145/335043.335045. URL http://portal.acm.org/citation.cfm?id=335043.335045&coll=GUIDE&dl=GUIDE&CFID=47964849&CFTOKEN=55429101.

[12] Andre DeHon. Balancing interconnect and computation in a reconfigurable computing array (or, why you don't really want 100% LUT utilization). In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 69–78, Monterey, California, United States, 1999. ACM. ISBN 1-58113-088-0. doi: 10.1145/296399.296431. URL http://portal.acm.org/citation.cfm?id=296431.

[13] Wei Mark Fang and Jonathan Rose. Modeling routing demand for early-stage FPGA architecture development. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 139–148, Monterey, California, USA, 2008. ACM. ISBN 978-1-59593-934-0. doi: 10.1145/1344671.1344694. URL http://portal.acm.org/citation.cfm?id=1344671.1344694.

[14] A.E. Gamal. Two-dimensional stochastic model for interconnections in master slice integrated circuits. *IEEE Transactions on Circuits and Systems*, 28 (2):127–138, 1981. ISSN 0098-4094. URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?&arNumber=1084958.

[15] Chi-Chou Kao and Yen-Tai Lai. An efficient algorithm for finding the minimal-area FPGA technology mapping. *ACM Transactions on Design Automation of Electronic Systems*, 10(1):168–186, 2005. doi: 10.1145/1044111.1044121. URL `http://portal.acm.org/citation.cfm?id=1044111.1044121&coll=GUIDE&dl=GUIDE&CFID=47964849&CFTOKEN=55429101`.

[16] Julien Lamoureux. On the interaction between Power-Aware Computer-Aided design algorithms for Field-Programable gate arrays. Master's thesis, University of British Columbia, June 2003. URL `http://www.ece.ubc.ca/~stevew/papers/pdf/lamoureux_masc.pdf`.

[17] Guy G. F. Lemieux, Stephen D. Brown, and Daniel Vranesic. On two-step routing for FPGAS. In *International Symposium on Physical Design*, pages 60–66, Napa Valley, California, United States, 1997. ACM. ISBN 0-89791-927-0. doi: 10.1145/267665.267682. URL `http://portal.acm.org/citation.cfm?id=267665.267682&coll=GUIDE&dl=GUIDE&CFID=48052556&CFTOKEN=40588312`.

[18] D. Leong and G. Lemieux. RePlace: an incremental placement algorithm for field-programmable gate arrays. In *International Conference on Field Programmable Logic and Applications*, 2009.

[19] Jai-Ming Lin, Song-Ra Pan, and Yao-Wen Chang. Graph matching-based algorithms for array-based FPGA segmentation design and routing. In *Asia and South Pacific Design Automation Conference*, pages 851–854, Kitakyushu, Japan, 2003. ACM. ISBN 0-7803-7660-9. doi: 10.1145/1119772.1119959. URL `http://portal.acm.org/citation.cfm?id=1119772.1119959&coll=GUIDE&dl=GUIDE&CFID=48052556&CFTOKEN=40588312`.

[20] L. McMurchie and C. Ebeling. PathFinder: a negotiation-based performance-driven router for FPGAs. In *ACM International Symposium on Field-Programmable Gate Arrays*, pages 111–117, 1995. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1377269`.

[21] MCNC. LGSynth93 benchmark suite. Technical report, Microelectronics Centre of North Carolina, 1993.

[22] M. Palczewski. Plane parallel a maze router and its application to FPGAs. In *ACM/IEEE Design Automation Conference*, pages 691–697, Anaheim, California, United States, 1992. IEEE Computer Society Press. ISBN 0-89791-516-X. URL `http://portal.acm.org/citation.cfm?id=149679`.

[23] A. Pandit and A. Akoglu. Net length based routability driven packing. In *International Conference on Field-Programmable Technology*, pages 225–232, 2007. doi: {10.1109/FPT.2007.4439253}. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4439253`.

[24] G. Parthasarathy, M. Marek-Sadowska, Arindam Mukherjee, and Amit Singh. Interconnect complexity-aware FPGA placement using Rent's rule. In *International Workshop on System-Level Interconnect Prediction*, pages 115–121, Sonoma, California, United States, 2001. ACM. ISBN 1-58113-315-4. doi: 10.1145/368640.368806. URL `http://portal.acm.org/citation.cfm?id=368806`.

[25] J. Rose. Parallel global routing for standard cells. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(10):1085–1095, 1990. ISSN 0278-0070. doi: 10.1109/43.62733. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=62733`.

[26] Amit Singh and Malgorzata Marek-Sadowska. Efficient circuit clustering for area and power reduction in FPGAs. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 59–66, Monterey, California, USA, 2002. ACM. ISBN 1-58113-452-5. doi: 10.1145/503048.503058. URL `http://portal.acm.org/citation.cfm?id=503058`.

[27] Russell Tessier and Heather Giza. Balancing logic utilization and area efficiency in FPGAs. In *International Conference on Field-Programmable Logic and Applications*, pages 535–544. Springer-Verlag, 2000. ISBN 3-540-67899-9. URL `http://portal.acm.org/citation.cfm?id=739548`.

[28] M. Tom. Channel width reduction techniques for System-on-Chip circuits in field-programmable gate arrays. Master's thesis, University of British Columbia, April 2006.

[29] Marvin Tom, David Leong, and Guy Lemieux. Un/DoPack: re-clustering of large system-on-chip designs with interconnect variation for low-cost FPGAs. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 680–687, San Jose, California, 2006. ACM. ISBN 1-59593-389-1. doi: 10.1145/1233501.1233643. URL `http://portal.acm.org/citation.cfm?id=1233643`.

[30] Mark Yamashita. A combined clustering and placement algorithm for FPGAs. Master's thesis, University of British Columbia, November 2007. URL `http://www.ece.ubc.ca/~lemieux/publications/yamashita-masc2007.pdf`.

[31] David Yeager, Darius Chiu, and Guy Lemieux. Congestion estimation and localization in FPGAs: a visual tool for interconnect prediction. In *In-*

ternational Workshop on System Level Interconnect Prediction, pages 33–40, Austin, Texas, USA, 2007. ACM. ISBN 978-1-59593-622-6. doi: 10.1145/1231956.1231963. URL http://portal.acm.org/citation.cfm?id=1231963.

[32] Yue Zhuo, Hao Li, and S.P. Mohanty. A congestion driven placement algorithm for FPGA synthesis. In *International Conference on Field Programmable Logic and Applications*, pages 1–4, 2006. doi: {10.1109/FPL.2006.311290}. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4101052.

Appendix A

Baseline Un/DoPack

CW	Runtime	CLBs	CP	Area
100	2867	3148	5.06E-08	1.82E+08
95	6001	3596	5.26E-08	2.00E+08
90	6506	3911	5.53E-08	2.15E+08
85	9518	4449	5.70E-08	2.37E+08
80	13616	4863	6.15E-08	2.51E+08
75	19318	5445	5.90E-08	2.75E+08
70	29127	6392	5.96E-08	3.14E+08
65	52205	7290	6.46E-08	3.52E+08

Table A.1: Baseline Un/DoPack - Stdev0

CW	Runtime	CLBs	CP	Area
105	3286	3157	5.28E-08	1.87E+08
100	3085	3157	5.28E-08	1.82E+08
95	3470	3290	5.74E-08	1.85E+08
90	3163	3298	5.34E-08	1.82E+08
85	5599	3683	6.03E-08	1.97E+08
80	7888	4220	5.67E-08	2.18E+08
75	10330	4656	5.87E-08	2.37E+08
70	17390	5351	5.75E-08	2.65E+08
65	26359	6029	5.91E-08	2.90E+08
60	35047	6867	6.31E-08	3.21E+08

Table A.2: Baseline Un/DoPack - Stdev002

CW	Runtime	CLBs	CP	Area
100	2977	3302	5.79E-08	1.90E+08
95	3136	3273	5.98E-08	1.85E+08
90	3491	3298	6.01E-08	1.82E+08
85	4113	3409	6.14E-08	1.83E+08
80	7093	3961	5.87E-08	2.04E+08
75	13240	5014	5.78E-08	2.53E+08
70	16985	5196	5.68E-08	2.58E+08
65	31989	6195	5.89E-08	2.98E+08

Table A.3: Baseline Un/DoPack - Stdev004

CW	Runtime	CLBs	CP	Area
95	2772	3139	5.29E-08	1.78E+08
90	3045	3139	5.50E-08	1.74E+08
85	5943	3889	5.57E-08	2.09E+08
80	7908	4090	5.65E-08	2.11E+08
75	12817	4916	5.90E-08	2.50E+08
70	20791	6067	6.02E-08	2.98E+08
65	30157	6443	5.98E-08	3.11E+08

Table A.4: Baseline Un/DoPack - Stdev006

CW	Runtime	CLBs	CP	Area
125	3548	3151	5.21E-08	2.02E+08
120	4178	3286	5.60E-08	2.05E+08
115	3600	3282	5.92E-08	2.02E+08
110	4399	3289	5.27E-08	1.97E+08
105	5174	3560	5.78E-08	2.08E+08
100	4645	3412	5.48E-08	1.96E+08
95	11595	4426	5.72E-08	2.47E+08
90	12227	4281	6.13E-08	2.35E+08
85	17837	4832	5.94E-08	2.58E+08
80	23261	4999	5.78E-08	2.58E+08
75	35152	5593	5.99E-08	2.83E+08
70	50869	6654	6.06E-08	3.28E+08

Table A.5: Baseline Un/DoPack - Clone

CW	Runtime	CLBs	CP	Area
165	3544	3152	5.81E-08	2.35E+08
155	3105	3152	5.94E-08	2.26E+08
150	4100	3304	5.76-08	2.31E+08
145	4154	3307	5.66E-08	2.27E+08
140	4309	3282	5.92E-08	2.22E+08
135	5236	3460	6.39E-08	2.27E+08
130	6200	3469	6.00E-08	2.24E+08
125	6618	3434	6.10E-08	2.18E+08
120	9696	3573	6.08E-08	2.21E+08
115	9348	3665	6.27E-08	2.24E+08
110	15258	3839	6.21E-08	2.28E+08
105	15797	3952	6.33E-08	2.30E+08
100	18072	4015	6.50E-08	2.31E+08
95	30961	4813	6.49E-08	2.69E+08
90	31815	4879	6.48E-08	2.66E+08
85	39060	5170	6.26E-08	2.75E+08

Table A.6: Baseline Un/DoPack - Stdev010

CW	Runtime	CLBs	CP	Area
155	4131	3304	5.93E-08	2.35E+08
150	3588	3163	5.74E-08	2.22E+08
145	4081	3163	5.82E-08	2.19E+08
140	4918	3313	5.84E-08	2.23E+08
135	6957	3600	5.96E-08	2.36E+08
130	7994	3586	5.87E-08	2.31E+08
125	9816	3750	6.22E-08	2.39E+08
120	13155	3943	6.09E-08	2.44E+08
115	18225	4501	6.28E-08	2.76E+08
110	28230	4861	6.35E-08	2.89E+08
105	30233	4628	6.41E-08	2.73E+08
100	39021	5095	6.12E-08	2.92E+08

Table A.7: Baseline Un/DoPack - Stdev012

Appendix B

Fine-Grained Un/DoPack

CW	Runtime	CLBs	CP	Area
100	3131	3148	5.06E-08	1.82E+08
95	5611	3223	5.28E-08	1.80E+08
90	10794	3404	5.42E-08	1.88E+08
85	19360	3618	5.46E-08	1.95E+08
80	47615	4236	5.89E-08	2.21E+08
75	62504	4410	6.17E-08	2.24E+08
70	106495	5135	6.05E-08	2.53E+08
65	203021	5817	6.09E-08	2.81E+08

Table B.1: Fine-Grained Un/DoPack - Stdev0

CW	Runtime	CLBs	CP	Area
105	2572	3157	5.28E-08	1.87E+08
100	2853	3157	5.28E-08	1.82E+08
95	3294	3178	5.55E-08	1.79E+08
90	5124	3253	5.83E-08	1.80E+08
85	6820	3286	5.65E-08	1.77E+08
80	21225	3662	5.78E-08	1.90E+08
75	31006	3853	5.76E-08	1.97E+08
70	55206	4383	5.60E-08	2.17E+08
65	92211	4834	6.02E-08	2.33E+08
60	203822	5931	6.05E-08	2.80E+08

Table B.2: Fine-Grained Un/DoPack - Stdev002

CW	Runtime	CLBs	CP	Area
100	3178	3169	5.68E-08	1.83E+08
95	3536	3188	5.83E-08	1.79E+08
90	4911	3220	5.58E-08	1.76E+08
85	5970	3261	5.70E-08	1.76E+08
80	16577	3547	5.52E-08	1.84E+08
75	19323	3600	5.62E-08	1.82E+08
70	62945	4558	5.90E-08	2.25E+08
65	96523	5020	5.63E-08	2.41E+08

Table B.3: Fine-Grained Un/DoPack - Stdev004

CW	Runtime	CLBs	CP	Area
95	2749	3139	5.29E-08	1.78E+08
90	2639	3139	5.50E-08	1.74E+08
85	8399	3356	5.57E-08	1.79E+08
80	18600	3682	5.61E-08	1.91E+08
75	35314	4031	5.63E-08	2.05E+08
70	50198	4447	5.73E-08	2.19E+08
65	80474	4812	6.10E-08	2.33E+08

Table B.4: Fine-Grained Un/DoPack - Stdev006

CW	Runtime	CLBs	CP	Area
125	3074	3151	5.22E-08	2.02E+08
120	3343	3172	5.37E-08	1.98E+08
115	3386	3172	5.48E-08	1.95E+08
110	5849	3229	5.33E-08	1.92E+08
105	6670	3262	5.41E-08	1.93E+08
100	10818	3348	5.42E-08	1.91E+08
95	15808	3441	5.48E-08	1.92E+08
90	24808	3616	5.55E-08	2.00E+08
85	44691	3870	6.07E-08	2.08E+08
80	87193	4359	5.74E-08	2.28E+08
75	112947	4758	5.64E-08	2.40E+08

Table B.5: Fine-Grained Un/DoPack - Clone

CW	Runtime	CLBs	CP	Area
165	3202	3152	5.81E-08	2.35E+08
155	3237	3152	5.94E-08	2.26E+08
150	4026	3166	6.10E-08	2.22E+08
145	5523	3183	6.01E-08	2.19E+08
140	8736	3236	6.07E-08	2.17E+08
135	9679	3259	5.89E-08	2.17E+08
130	11838	3284	6.43E-08	2.14E+08
125	18518	3371	6.05E-08	2.16E+08
120	18326	3379	6.28E-08	2.12E+08
115	26796	3441	6.18E-08	2.10E+08
110	40918	3613	6.26E-08	2.18E+08
105	60074	3639	6.39E-08	2.14E+08
100	74675	3856	6.38E-08	2.22E+08
95	107440	4181	6.77E-08	2.33E+08
90	148997	4348	6.59E-08	2.37E+08

Table B.6: Fine-Grained Un/DoPack - Stdev010

CW	Runtime	CLBs	CP	Area
155	4323	3184	5.83E-08	2.27E+08
150	3346	3163	5.75E-08	2.22E+08
145	3644	3163	5.82E-08	2.19E+08
140	6043	3196	5.92E-08	2.16E+08
135	8672	3251	6.07E-08	2.17E+08
130	15415	3356	5.83E-08	2.16E+08
125	23355	3418	6.05E-08	2.18E+08
120	37426	3558	6.46E-08	2.21E+08
115	56216	3781	6.11E-08	2.31E+08
110	66307	3969	6.21E-08	2.35E+08
105	103231	4233	6.16E-08	2.50E+08
100	149969	4412	6.54E-08	2.53E+08

Table B.7: Fine-Grained Un/DoPack - Stdev012

Appendix C

Multiregion Un/DoPack

CW	Runtime	CLBs	CP	Area
100	2567	3148	5.06E-08	1.82E+08
95	4438	3434	5.26E-08	1.92E+08
90	6894	3667	5.87E-08	2.01E+08
85	9903	5148	5.76E-08	2.74E+08
80	12917	6095	5.84E-08	3.17E+08
75	10678	6096	5.98E-08	3.10E+08
70	8682	6102	5.65E-08	3.02E+08
65	10746	7174	5.86E-08	3.45E+08

Table C.1: Multiregion Un/DoPack - Stdev0

CW	Runtime	CLBs	CP	Area
105	2517	3157	5.28E-08	1.87E+08
100	3143	3157	5.28E-08	1.82E+08
95	2719	3195	5.69E-08	1.79E+08
90	3630	3284	5.56E-08	1.81E+08
85	3461	3455	5.47E-08	1.85E+08
80	8710	4177	5.67E-08	2.16E+08
75	7164	4531	6.17E-08	2.30E+08
70	4620	4695	5.96E-08	2.32E+08
65	6528	6328	6.09E-08	3.05E+08
60	10319	8812	5.99E-08	4.11E+08

Table C.2: Multiregion Un/DoPack - Stdev002

CW	Runtime	CLBs	CP	Area
100	3571	3171	5.58E-08	1.83E+08
95	4066	3217	5.72E-08	1.80E+08
90	4784	3355	5.89E-08	1.83E+08
85	4233	3304	5.72E-08	1.77E+08
80	8429	4128	5.83E-08	2.15E+08
75	8477	4968	6.02E-08	2.52E+08
70	9832	6438	5.76E-08	3.18E+08
65	7965	7167	6.23E-08	3.45E+08

Table C.3: Multiregion Un/DoPack - Stdev004

CW	Runtime	CLBs	CP	Area
95	3142	3139	5.29E-08	1.78E+08
90	2945	3139	5.50E-08	1.74E+08
85	8741	3713	5.86E-08	1.98E+08
80	11644	5385	5.94E-08	2.79E+08
75	9591	5407	6.62E-08	2.74E+08
70	9060	6561	5.74E-08	3.22E+08
65	5008	6824	6.40E-08	3.29E+08

Table C.4: Multiregion Un/DoPack - Stdev006

CW	Runtime	CLBs	CP	Area
125	3616	3151	5.22E-08	2.02E+08
120	3056	3165	5.39E-08	1.98E+08
115	4217	3206	5.37E-08	1.96E+08
110	5505	3256	5.56E-08	1.97E+08
105	6832	3384	5.70E-08	2.00E+08
100	4312	3309	5.55E-08	1.90E+08
95	6314	3620	5.79E-08	2.04E+08
90	9575	4101	5.63E-08	2.26E+08
85	7616	4634	5.81E-08	2.49E+08
80	11451	5539	6.53E-08	2.87E+08
75	8669	6159	6.26E-08	3.12E+08
70	16623	7240	6.23E-08	3.58E+08

Table C.5: Multiregion Un/DoPack - Clone

CW	Runtime	CLBs	CP	Area
165	2834	3152	5.81E-08	2.35E+08
155	3488	3152	5.94E-08	2.26E+08
150	3792	3182	5.85E-08	2.23E+08
145	6099	3222	6.13E-08	2.21E+08
140	11691	3227	5.89E-08	2.16E+08
135	5515	3257	5.98E-08	2.17E+08
130	6878	3251	6.16E-08	2.13E+08
125	8846	3427	6.11E-08	2.18E+08
120	8932	3537	6.36E-08	2.20E+08
115	7047	3576	6.06E-08	2.18E+08
110	9549	3771	6.40E-08	2.26E+08
105	9432	3852	6.32E-08	2.28E+08
100	10711	4284	6.12E-08	2.46E+08
95	10807	4492	6.34E-08	2.53E+08
90	14656	5300	6.42E-08	2.89E+08
85	13677	6350	6.52E-08	3.38E+08

Table C.6: Multiregion Un/DoPack - Stdev010

CW	Runtime	CLBs	CP	Area
155	4589	3168	5.86E-08	2.27E+08
150	3402	3163	5.75E-08	2.22E+08
145	3755	3163	5.82E-08	2.19E+08
140	7819	3294	6.24E-08	2.23E+08
135	6325	3331	5.93E-08	2.19E+08
130	17459	3595	6.07E-08	2.32E+08
125	17165	3793	6.22E-08	2.41E+08
120	13467	4006	6.42E-08	2.50E+08
115	14687	4402	6.23E-08	2.69E+08
110	14980	4569	6.27E-08	2.72E+08
105	15235	4855	6.96E-08	2.83E+08
100	28027	5266	6.51E-08	3.01E+08

Table C.7: Multiregion Un/DoPack - Stdev012

Appendix D

Budgeted Multiregion Un/DoPack

CW	Runtime	CLBs	CP	Area
100	3018	3148	5.06E-08	1.82E+08
95	3559	3248	5.29E-08	1.81E+08
90	5284	3598	5.29E-08	1.96E+08
85	7387	3833	5.37E-08	2.04E+08
80	11016	4352	5.43E-08	2.24E+08
75	14184	4366	5.49E-08	2.23E+08
70	24214	5472	5.96E-08	2.69E+08
65	34074	5503	5.70E-08	2.78E+08

Table D.1: BMR Un/DoPack - Stdev0

CW	Runtime	CLBs	CP	Area
105	2463	3157	5.28E-08	1.87E+08
100	2342	3157	5.28E-08	1.82E+08
95	3664	3247	5.67E-08	1.81E+08
90	2772	3249	5.48E-08	1.77E+08
85	4347	3276	5.61E-08	1.77E+08
80	6258	3708	5.64E-08	1.92E+08
75	10603	4220	5.64E-08	2.13E+08
70	14223	4621	5.57E-08	2.27E+08
65	18115	5040	5.90E-08	2.42E+08
60	28257	5743	6.52E-08	1.69E+08

Table D.2: BMR Un/DoPack - Stdev002

CW	Runtime	CLBs	CP	Area
100	2875	3246	5.69E-08	1.85E+08
95	3699	3171	5.76E-08	1.79E+08
90	3890	3363	5.76E-08	1.83E+08
85	3986	3478	5.51E-08	1.85E+08
80	6901	3618	5.82E-08	1.89E+08
75	10066	4115	6.13E-08	2.10E+08
70	15015	4898	6.38E-08	2.41E+08
65	19572	5023	5.80E-08	2.42E+08

Table D.3: BMR Un/DoPack - Stdev004

CW	Runtime	CLBs	CP	Area
95	2354	3139	5.29E-08	1.78E+08
90	2357	3139	5.50E-08	1.74E+08
85	6924	3621	5.56E-08	1.95E+08
80	8953	3967	5.43E-08	2.05E+08
75	8036	3765	5.51E-08	1.92E+08
70	13754	4619	5.67E-08	2.27E+08
65	22104	5141	5.63E-08	2.48E+08

Table D.4: BMR Un/DoPack - Stdev006

CW	Runtime	CLBs	CP	Area
125	2864	3151	5.22E-08	2.02E+08
120	3349	3249	5.22E-08	2.01E+08
115	3756	3248	5.57E-08	1.97E+08
110	4786	3362	5.53E-08	2.00E+08
105	5155	3478	5.89E-08	2.03E+08
100	5267	3475	5.61E-08	1.98E+08
95	8672	3599	5.60E-08	2.00E+08
90	9089	3835	6.08E-08	2.09E+08
85	12550	4221	5.62E-08	2.25E+08
80	19763	4896	5.94E-08	2.52E+08
75	23908	5327	5.99E-08	2.69E+08
70	32674	5761	5.85E-08	2.83E+08

Table D.5: BMR Un/DoPack - Clone

CW	Runtime	CLBs	CP	Area
165	3553	3152	5.81E-08	2.35E+08
155	3022	3152	5.94E-08	2.26E+08
150	4665	3247	6.17E-08	2.25E+08
145	5587	3286	5.87E-08	2.27E+08
140	5473	3310	6.31E-08	2.23E+08
135	5607	3246	6.18E-08	2.13E+08
130	6316	3360	6.35E-08	2.16E+08
125	7346	3474	6.17E-08	2.19E+08
120	8355	3480	6.48E-08	2.15E+08
115	9190	3384	6.06E-08	2.08E+08
110	11261	3499	6.05E-08	2.11E+08
105	14383	3714	6.13E-08	2.16E+08
100	16032	3837	6.62E-08	2.19E+08
95	23491	4217	6.72E-08	2.34E+08
90	28701	4487	6.59E-08	2.44E+08
85	41871	4517	6.23E-08	2.43E+08

Table D.6: BMR Un/DoPack - Stdev010

CW	Runtime	CLBs	CP	Area
155	4559	3230	5.85E-08	2.29E+08
150	3295	3163	5.75E-08	2.22E+08
145	3423	3163	5.82E-08	2.19E+08
140	4541	3248	5.72E-08	2.17E+08
135	5958	3358	5.90E-08	2.20E+08
130	8241	3478	5.94E-08	2.24E+08
125	8883	3475	6.20E-08	2.19E+08
120	13624	3840	6.06E-08	2.37E+08
115	17338	3957	6.19E-08	2.40E+08
110	20845	4093	6.05E-08	2.43E+08
105	31227	4486	6.11E-08	2.61E+08
100	41038	4896	6.61E-08	2.78E+08

Table D.7: BMR Un/DoPack - Stdev012

Appendix E

Congestion-Driven Multiregion Un/DoPack

CW	Runtime	CLBs	CP	Area
100	2758	3148	5.06E-08	1.82E+08
95	4464	3254	5.31E-08	1.84E+08
90	5133	3519	5.50E-08	1.94E+08
85	3644	3515	5.16E-08	1.89E+08
80	7905	4794	6.12E-08	2.49E+08
75	8154	4392	5.42E-08	2.24E+08
70	11176	6011	5.82E-08	2.96E+08
65	9519	6457	5.63E-08	3.12E+08

Table E.1: CMR Un/DoPack - Stdev0

CW	Runtime	CLBs	CP	Area
105	2940	3157	5.28E-08	1.87E+08
100	2641	3157	5.28E-08	1.82E+08
95	3081	3203	5.66E-08	1.79E+08
90	3615	3243	5.53E-08	1.77E+08
85	4361	3437	5.53E-08	1.84E+08
80	6365	4250	5.75E-08	2.21E+08
75	5382	4129	5.73E-08	2.10E+08
70	6155	4767	5.48E-08	2.37E+08
65	6848	5017	5.99E-08	2.41E+08
60	8921	6115	6.43E-08	2.88E+08

Table E.2: CMR Un/DoPack - Stdev002

CW	Runtime	CLBs	CP	Area
100	3571	3171	5.58E-08	1.83E+08
95	2953	3171	5.76E-08	1.79E+08
90	4320	3390	6.24E-08	1.87E+08
85	4734	3610	5.81E-08	1.95E+08
80	6107	3815	6.00E-08	1.97E+08
75	6027	4492	5.85E-08	2.29E+08
70	6276	4567	5.78E-08	2.26E+08
65	7227	5163	6.22E-08	2.48E+08

Table E.3: CMR Un/DoPack - Stdev004

CW	Runtime	CLBs	CP	Area
95	2518	3139	5.29E-08	1.78E+08
90	2633	3139	5.50E-08	1.74E+08
85	5669	3771	5.58E-08	2.02E+08
80	4295	3718	5.53E-08	1.92E+08
75	7822	4368	5.70E-08	2.23E+08
70	6953	4737	5.60E-08	2.33E+08
65	10167	6615	6.05E-08	3.19E+08

Table E.4: CMR Un/DoPack - Stdev006

CW	Runtime	CLBs	CP	Area
125	2770	3151	5.22E-08	2.02E+08
120	3493	3195	5.38E-08	1.99E+08
115	3375	3174	5.63E-08	1.95E+08
110	4636	3239	5.41E-08	1.93E+08
105	4953	3419	5.41E-08	2.01E+08
100	4968	3604	5.98E-08	2.08E+08
95	5499	3757	5.77E-08	2.11E+08
90	7454	3855	5.81E-08	2.13E+08
85	7279	4400	5.70E-08	2.36E+08
80	7611	4581	6.03E-08	2.37E+08
75	8315	5120	6.12E-08	2.59E+08
70	10724	6511	6.12E-08	3.20E+08

Table E.5: CMR Un/DoPack - Clone

CW	Runtime	CLBs	CP	Area
165	0	0		0.00E+00
155	3835	3152	5.94E-08	2.26E+08
150	4638	3191	6.03E-08	2.23E+08
145	4933	3195	6.01E-08	2.20E+08
140	6033	3258	6.11E-08	2.22E+08
135	6333	3313	6.12E-08	2.19E+08
130	5683	3318	6.14E-08	2.15E+08
125	6167	3346	6.00E-08	2.12E+08
120	8045	3483	6.19E-08	2.19E+08
115	9024	3575	6.16E-08	2.18E+08
110	9552	3722	6.13E-08	2.24E+08
105	9127	3740	6.26E-08	2.21E+08
100	12048	4144	6.18E-08	2.38E+08
95	11672	4404	6.54E-08	2.47E+08
90	11766	4770	6.43E-08	2.63E+08
85	10788	5025	6.75E-08	2.67E+08

Table E.6: CMR Un/DoPack - Stdev010

CW	Runtime	CLBs	CP	Area
155	4897	3186	5.73E-08	2.27E+08
150	3568	3163	5.75E-08	2.22E+08
145	3901	3163	5.82E-08	2.19E+08
140	5579	3202	6.33E-08	2.16E+08
135	6569	3285	6.24E-08	2.18E+08
130	8228	3401	6.04E-08	2.22E+08
125	7529	3515	5.87E-08	2.24E+08
120	8830	3729	5.87E-08	2.34E+08
115	12320	4014	6.06E-08	2.46E+08
110	10783	4308	6.20E-08	2.57E+08
105	15059	4619	6.27E-08	2.69E+08
100	21536	5223	6.44E-08	3.00E+08

Table E.7: CMR Un/DoPack - Stdev012

Appendix F

CMR Un/DoPack with Congestion-Aware Placement

CW	Runtime	CLBs	CP	Area
95	11541	3148	5.88E-08	1.78E+08
90	14170	3368	5.65E-08	1.87E+08
85	16844	3740	5.95E-08	2.01E+08
80	21482	4113	6.24E-08	2.14E+08
75	32455	4230	6.38E-08	2.16E+08
70	39878	4686	7.01E-08	2.32E+08
65	42141	5302	6.57E-08	2.55E+08

Table F.1: CMR Un/DoPack (Congestion Aware Placement) - Stdev0

CW	Runtime	CLBs	CP	Area
105	11020	3157	5.70E-08	1.87E+08
100	10988	3157	5.68E-08	1.82E+08
95	11022	3157	6.03E-08	1.78E+08
90	10980	3157	5.70E-08	1.75E+08
85	16253	3304	5.62E-08	1.77E+08
80	21771	3588	6.20E-08	1.85E+08
75	38213	3754	6.04E-08	1.91E+08
70	43487	4265	6.79E-08	2.11E+08

Table F.2: CMR Un/DoPack (Congestion Aware Placement) - Stdev002

CW	Runtime	CLBs	CP	Area
100	10098	3148	5.91E-08	1.82E+08
95	9939	3148	5.91E-08	1.78E+08
90	13714	3263	5.81E-08	1.81E+08
85	14727	3387	5.74E-08	1.83E+08
80	19897	3569	6.04E-08	1.85E+08
75	24217	3606	6.03E-08	1.85E+08
70	40794	4095	5.82E-08	2.01E+08
65	42435	4617	6.03E-08	2.22E+08

Table F.3: CMR Un/DoPack (Congestion Aware Placement) - Stdev004

CW	Runtime	CLBs	CP	Area
95	10159	3139	6.48E-08	1.78E+08
90	10413	3139	6.48E-08	1.74E+08
85	12909	3279	5.96E-08	1.77E+08
80	17524	3912	6.62E-08	2.03E+08
75	20202	3859	6.90E-08	1.97E+08
70	23249	4340	6.77E-08	2.14E+08
65	40967	4987	6.70E-08	2.41E+08

Table F.4: CMR Un/DoPack (Congestion Aware Placement) - Stdev006

CW	Runtime	CLBs	CP	Area
125	10657	3151	6.19E-08	2.02E+08
120	10669	3151	6.22E-08	1.98E+08
115	10574	3151	6.24E-08	1.94E+08
110	10733	3151	6.35E-08	1.90E+08
105	12792	3252	6.46E-08	1.93E+08
100	13492	3316	5.98E-08	1.90E+08
95	15501	3384	6.39E-08	1.91E+08
90	20186	3825	6.69E-08	2.09E+08
85	25313	4204	6.70E-08	2.24E+08
80	32880	4244	6.72E-08	2.21E+08
75	42864	5226	6.70E-08	2.66E+08
70	44757	5429	6.93E-08	2.67E+08

Table F.5: CMR Un/DoPack (Congestion Aware Placement) - Clone

CW	Runtime	CLBs	CP	Area
165	11513	3152	6.85E-08	2.35E+08
155	11622	3152	6.85E-08	2.26E+08
150	11137	3152	6.94E-08	2.22E+08
145	11820	3152	6.98E-08	2.18E+08
140	11385	3152	7.00E-08	2.14E+08
135	16021	3297	6.45E-08	2.18E+08
130	15098	3339	6.91E-08	2.16E+08
125	17910	3382	6.77E-08	2.17E+08
120	16409	3447	7.18E-08	2.14E+08
115	17978	3579	7.47E-08	2.18E+08
110	19868	3777	7.14E-08	2.26E+08
105	20037	3826	7.37E-08	2.23E+08
100	23066	4175	8.02E-08	2.39E+08
95	21746	4316	8.01E-08	2.41E+08
90	30465	5035	7.71E-08	2.74E+08
85	39128	5369	7.84E-08	2.88E+08

Table F.6: CMR Un/DoPack (Congestion Aware Placement) - Stdev010

CW	Runtime	CLBs	CP	Area
155	13152	3163	6.97E-08	2.27E+08
150	13351	3163	6.98E-08	2.22E+08
145	13650	3163	6.98E-08	2.19E+08
140	13761	3163	6.95E-08	2.15E+08
135	19766	3278	7.95E-08	2.18E+08
130	22315	3355	8.05E-08	2.16E+08
125	20295	3393	7.70E-08	2.17E+08
120	27646	3661	7.73E-08	2.28E+08
115	28116	3761	7.79E-08	2.31E+08
110	31908	3995	8.22E-08	2.40E+08
105	35722	4391	8.03E-08	2.58E+08
100	38933	4720	8.47E-08	2.69E+08

Table F.7: CMR Un/DoPack (Congestion Aware Placement) - Stdev012