CHANNEL WIDTH REDUCTION TECHNIQUES FOR SYSTEM-ON-CHIP CIRCUITS IN FIELD-PROGRAMMABLE GATE ARRAYS

by

MARVIN TOM

B.A.Sc., Simon Fraser University, 2002

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

April 2006

ABSTRACT

Users of field-programmable gate arrays (FPGAs) typically measure the size of a FPGA by its logic capacity. If a design fits within the logic capacity limits of an FPGA, it is generally assumed that it must also be routable. To ensure this high routability, FPGA vendors typically over-design the routing network. Despite this over-design, there may still be circuits that remain un-routable in a given FPGA family. This thesis presents two new computer-aided design (CAD) tools, DHPack and Un/DoPack, that are able to route these un-routable circuits by trading off logic utilization for interconnect. DHPack uses the natural design hierarchy of the circuit to identify high congestion regions. For a set of benchmark circuits used in this thesis, DHPack is able to reduce channel width by 13% with a small area increase of 3%. DHPack can continue to decrease channel width by 29% with a larger area increase of 146%. Un/DoPack improves on DHPack by targeting hard channel width constraints without having to rely on the design hierarchy of the circuit to perform congestion estimation. For a set of benchmark circuits presented in this thesis, Un/DoPack can reduce channel width by 38% with an 18% penalty in critical path delay and 64% increase in area. The primary application of these tools is to make previously un-routable circuits routable by using an FPGA with more logic.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

Application Specific Integrated Circuit (ASIC):	An integrated circuit intended for a specific use rather than general-purpose use. Once manufactured, the logical function of an ASIC cannot be changed
Basic Logic Element (BLE):	Logic element in an FPGA composed of a K-input LUT and flip-flop.
Computer Aided Design (CAD) Tools:	Software automation tools to aid in the design of electrical systems.
Configurable Logic Block (CLB):	Logic element in an FPGA composed of 'N' BLEs.
CLB Depopulation:	The process of inserting empty BLEs into a circuit mapped to a FPGA to reduce the MRCW.
Design Hierarchy Pack (DHPack):	An FPGA channel width reduction tool which relies on the design hierarchy of the circuit to identify congestion regions.
Field Programmable Gate Array (FPGA):	An integrated circuit that can be programmed, erased and re-programmed again to implement digital logic functions.
Generate Netlist (GNL):	A synthetic benchmark generator [51].
Interconnect Resource Aware Clustering (iRAC):	The state of the art FPGA clustering algorithm for channel width reduction [46].
Intellectual Property Blocks (IP Blocks):	A reusable unit of logic, cells or layout of an integrated circuit. SoC designs are created by merging IP blocks that have been pre-designed and pre-tested.
Look Up Table (LUT):	An FPGA element capable of implementing any logical function of its inputs.

Meta Benchmark Circuits:	A synthetic benchmark circuit suite created by stitching together the 20 largest MCNC benchmark circuits.
Microelectronics Corporation of North Carolina (MCNC) Circuits:	A standard set of benchmark circuits used in the FPGA academic community [36].
Minimum Routable Channel Width (MRCW):	The minimum channel width an FPGA must have to route a given circuit.
Non-Recurring Engineering Fees (NRE):	The one time costs of product development. This often includes mask costs and costs of CAD tools in integrated circuit design.
SIS:	A logic synthesis package developed at the University of California at Berkeley which allows interactive optimization of sequential digital circuits.
System-on-Chip (SoC):	A design philosophy which integrates all the components of an electronic system into a single integrated circuit. A SoC design philosophy makes the design of complex systems simpler by merging together pre-existing and pre-tested circuit designs.
Stdev Benchmark Circuits:	A synthetic benchmark suite created by cloning the Meta benchmarks. Each circuit in this suite represents a circuit with a varying amount of interconnect variation.
T-VPack:	The most commonly used academic FPGA clustering algorithm.
Un/DoPack:	An FPGA channel width reduction tool that can target hard channel width constraints.
Versatile Place and Route (VPR):	The most commonly used academic FPGA place and route tool.

ACKNOWLEDGEMENTS

I would like to thank my academic advisor, Dr. Guy Lemieux for his technical advice throughout my Master's degree. I have learned a great deal about academia and the research world from Guy and I am grateful for his guidance and support.

I would also like to thank the members of the UBC System-on-Chip research group for making my stay an enjoyable one. In particular, I'd like to thank Edmund Lee, Anthony Yu, Victor Aken'Ova, Martin Ma, James Wu, Amit Kedia, Scott Chin and Nathalie Chan for the good times in the lab.

I am grateful for the use of WestGrid[1] computing resources. The types of experiments that I have performed in this thesis would not have been possible without Westgrid.

Finally, I would like to thank my family for all the encouragement and support over the past few years.

[1] Westgrid is funded in part by the Canada Foundation for Innovation, Alberta Innovation and Science, BC Advanced Education, and the participating research institutions. WestGrid equipment is provided by IBM, Hewlett Packard and SGI.

Chapter 1

1 INTRODUCTION

A field- programmable gate array (FPGA) is capable of implementing a large variety of digital logic applications. Typically, FPGAs can be programmed, erased and re-programmed again multiple times. An alternative to FPGAs are application specific integrated circuits (ASICs) which are designed to perform one specific function. ASICs provide much higher speed, density and power characteristics than FPGAs but require very large up-front costs and cannot be changed after the manufacturing process. FPGAs are generally slower, larger and consume more power than their ASIC counterparts, but offer faster time-to-market and are programmable in the field after the manufacturing process. Many digital logic applications would benefit from the high performance characteristics of an ASIC, but these applications don't have the manufacturing volume needed to justify the $10+ million in computer-aided design (CAD) tools, design and verification costs, and non-recurring engineering (NRE) fees. Because FPGAs are not subject to most of these up-front costs, they are very attractive to low-to-medium density logic and low-to-medium volume designs.

As FPGAs increase in capacity and capability, it has become common to offer separate low-cost and resource-rich families. For a similar number of logic elements, also known as configurable logic blocks (CLBs), the low-cost families often have less

embedded memory, embedded multipliers, and routing tracks. This is demonstrated by Table 1-1, where the low-cost Altera Cyclone family offers significant savings. Unfortunately, some designs may fit within the Cyclone logic and memory capacity limits but not within the routing capacity limits. This can be solved by switching to the resource-rich family (e.g. Altera Stratix) at ~4x the cost. Instead, it is preferable to stay in the low-cost family and use the same or next-larger device (at ~2x cost). To do this, the FPGA computer-aided design (CAD) tools must meet the device routing capacity by targeting a *hard channel width constraint*. Since interconnect use of a design varies spatially with placement, this can be done by spreading out regions of peak demand to use fewer routing tracks but more CLBs.

Altera Device	Logic Elements	Memory	Mult.	Routing	Cost
Cyclone 1C12	12,060	239,616	0	80	$56
Stratix 1S10	10,570	920,448	48	232	$190
Cyclone 1C20	20,060	294,448	0	80	$100
Stratix 1S20	18,460	1,669,249	80	232	$350

Table 1-1: Features and Costs of Two FPGA Families (from [2], [3], [19])

1.1 Motivation and Objectives

The **minimum routable channel width (MRCW)** of a circuit is defined as the smallest possible channel width a FPGA device must have in order to route that circuit. This thesis presents an algorithmic way of reducing the minimum routable channel width (MRCW) of a logic design by inserting whitespace in the form of empty look-up tables (LUTs) into congested areas. Whitespace is inserted by identifying a congested region of CLBs, unpacking the CLBs into its constituent basic

logic elements (BLEs), and then re-packing these BLEs into more CLBs so they are "less full" than before. This process of inserting whitespace into each CLB is called *depopulating.*

Note that it is possible to reduce the MRCW of a circuit through clustering. Traditional clustering algorithms, such as T-VPack [6], fully pack the clusters to minimize the total number of CLBs needed. However, DeHon [17] and Tessier [48] have shown that the channel width needs of a circuit can be decreased by packing fewer BLEs into each CLB. The resulting "under-utilization" of CLBs is known as depopulation.

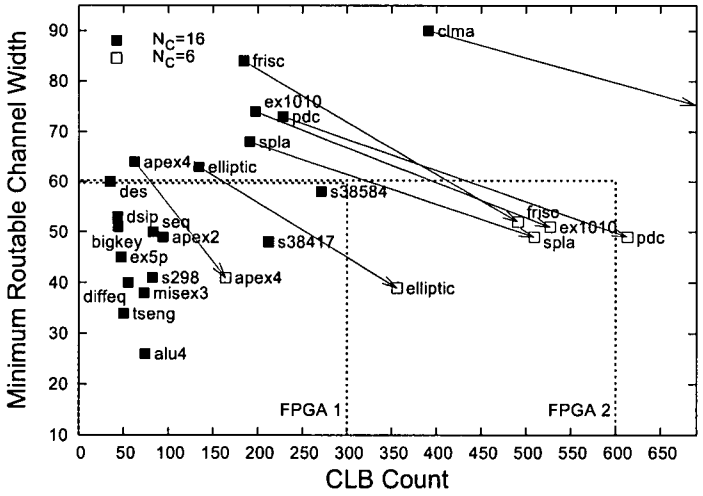

Figure 1-1: Channel Width / CLB Count Tradeoff

To see how depopulation works, consider the two large dashed boxes in Figure 1-1 representing the logic and routing capacities of two FPGA devices. These FPGAs contain 16 BLEs per CLB and 60 wiring tracks per routing channel. The MRCW of 20 MCNC benchmark circuits [36] after clustering (T-VPack [6]) and routing (VPR [6])

are shown. Notice that circuits with similar CLB counts can require vastly different channel widths (varying from 25 to 65). Similar results for industrial benchmarks are shown in [34].

In Figure 1-1, FPGA 1 contains 300 CLBs and can implement all circuits inside its box. In comparison, FPGA 2 has 600 CLBs and the same channel-width constraint of 60 because it is based on the same layout tile. Even though it is larger, FPGA 2 is incapable of realizing any circuits that require a channel width greater than 60, e.g. apex4 or elliptic. After depopulation (limiting to 6 BLEs per CLB), apex4's MRCW shrinks from 62 to 41 tracks. Although the CLB count increases, it still fits into FPGA 1. More importantly, apex4 now has a viable, routed solution. Similarly, some circuits like elliptic can be made to fit FPGA 2.

The problem with depopulation is that it quickly leads to an inflated CLB count. In the example, circuits pdc and clma are too large for FPGA 2. They must be depopulated less to meet the CLB constraint as well. What is needed is a way to depopulate only the routing-congested regions of a circuit so CLB count is inflated as little as possible. Such an approach is important for fitting large System-on-Chip designs onto modern FPGAs.

1.2 Contributions

This thesis presents two FPGA CAD tools that can depopulate an FPGA design to target channel width constraints. The first tool, DHPack, relies on the design hierarchy of the design to detect areas of congestion. Results of this work have been

published at the Design Automation Conference (DAC 2005) [49]. The second tool, Un/DoPack, is an iterative tool designed to target **hard** channel width constraints. A paper based on this work has been submitted to DAC 2006 [50]. The primary application of these tools is to reduce the channel width requirements of a circuit so that it can be mapped to a channel-width constrained FPGA. Rather than depopulate the entire circuit, which would inflate area rather quickly, the tools depopulate smaller regions (possibly entire IP blocks) that are interconnect-intensive.

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 presents an overview of modern mesh based FPGA architectures and the state of the art CAD tools to map circuits to these FPGAs. It also includes some discussion on previous techniques to reduce MRCW. Chapter 3 compares two basic depopulation approaches for channel width reduction. Chapter 4 presents two benchmark suites (Meta and Stdev) that mimic system-on-chip (SoC) designs and discusses the benchmark circuit characteristics that are important for channel width reduction. Chapter 5 presents the FPGA CAD tool DHPack which uses the natural design hierarchy of the circuit to identify high congestion regions. Chapter 6 presents the FPGA CAD tool Un/DoPack which iteratively depopulates circuits to meet hard channel width constraints. Finally, some conclusions are provided in Chapter 7 along with some possible future work.

Chapter 2

2 BACKGROUND

This chapter begins with an overview of modern FPGA architectures. The two most typical FPGA architectures are mesh-based and hierarchical. Since industrial FPGA vendors typically use mesh-based structures, the architectures and CAD tools discussed in this thesis will only target mesh-based FPGAs. An overview of the current state-of-the-art CAD algorithms that map digital circuits into FPGAs is then provided. The FPGA CAD flow can be split into 5 steps: synthesis, technology mapping, clustering, placement and routing. A survey of the most commonly used tools for each of these 5 steps is provided. The chapter concludes with a discussion on previous methods to reduce MRCW.

2.1 FPGA Architecture

A commercial FPGA family consists of a number of devices, each with a different logic capacity. Figure 2-1 illustrates the logic resources: CLBs and BLEs. A basic logic element (BLE) is composed of a K-input look-up table (LUT) and flip-flop. A K-input LUT has one dedicated output and is capable of implementing any Boolean function of its K-inputs. Logic capacity of an FPGA is measured by the number of BLEs. Alternatively, it can be measured by the number of CLBs, or

configurable logic blocks, which are simply fixed-size clusters of BLEs. Since mesh-based FPGAs are typically laid out in a 2-dimensional structure, device logic capacity can also be expressed by the logical dimensions of the CLB array.

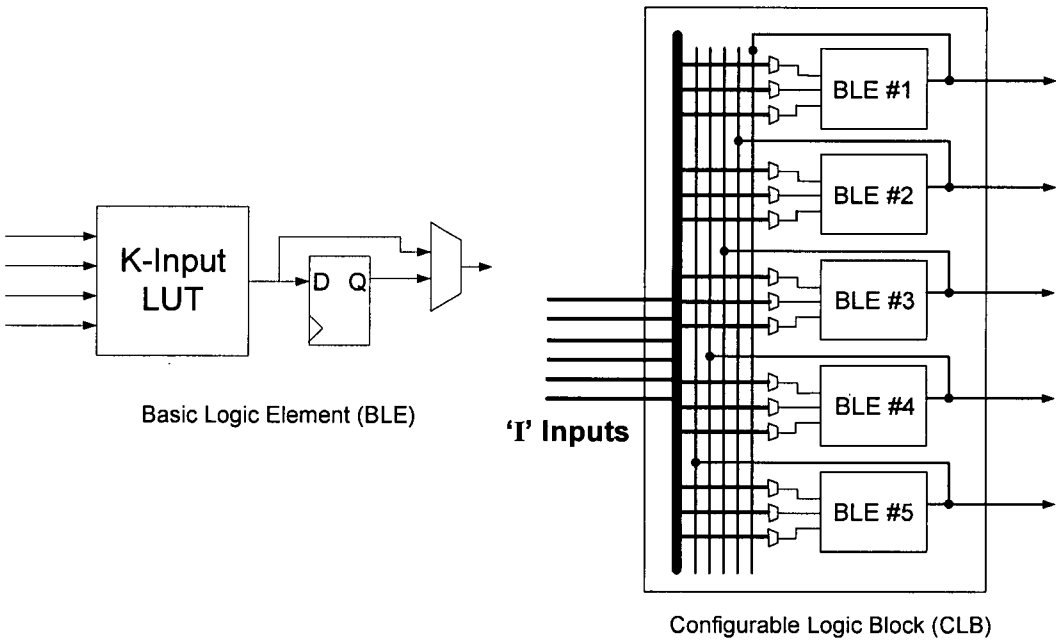

Figure 2-1: BLE and CLB

The logic elements in an FPGA are connected through a mesh based programmable interconnect network. A typical mesh based FPGA architecture similar to [2] and [53] is given in Figure 2-2.

The **channel width** of a mesh based FPGA architecture is defined by the number of routing tracks running in each horizontal and vertical channel. In Figure 2-2, the channel width is 4 since there are 4 tracks in each horizontal and vertical channel. This channel width is fixed across an entire FPGA family. The reason it is fixed is that larger sized FPGAs in the same family are created by placing more tiles

on a larger sized die. Since the channel width is a fixed feature on a tile, the inclusion of more tiles has no effect on the channel width of a family. For example, the Altera Cyclone device contains five different options in terms of logic capacity. This is demonstrated in Table 2-1. However each of these devices contains the same channel width constraint of 80 routing tracks per channel.

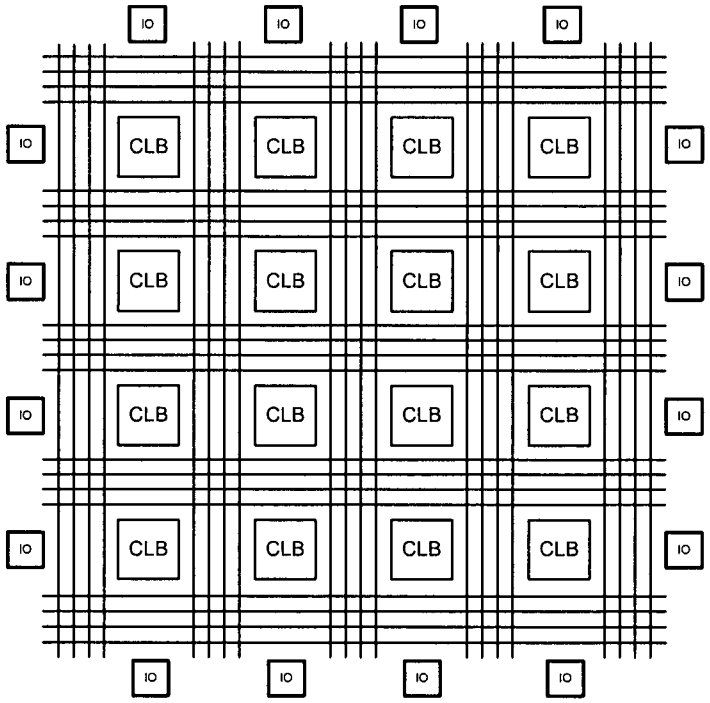

Figure 2-2: Mesh Based FPGA Architecture

Altera Device	EP1C3	EP1C4	EP1C6	EP1C12	EP1C20
Number of LEs	2,910	4,000	5,980	12,060	20,060
Number of Routing Tracks	80	80	80	80	80

Table 2-1: Altera Cyclone Size Options (from [2])

The LUT size, number of BLEs in each CLB and the number of inputs per cluster vary across many different vendors. For all of the experiments performed in the

remainder of this thesis, an FPGA architecture based on the parameters given below is used unless otherwise specified. Note that the channel width of the FPGA is left as a variable. The CAD tools described in this thesis attempt to find the minimum possible channel width needed to route a circuit.

- LUT Size (K)= 6
- Cluster Size (N) = 16
- Number of Inputs Per Cluster (I) = 51 = k/2*(N+1) (from [1])
- Length of Wires (L) = 4
- Switch Block Type (SB_{type}) = Subset
- C-Block Input Connectivity (F_{cin}) = 0.366
- C-Block Output Connectivity (F_{cout}) = 0.1
- C-Block I/O Pad Connectivity (F_{cpad}) = 1
- Fully Buffered Switches
- I/O Ratio = Minimum value to ensure circuit is logic limited
- Process Parameters = 0.18μm TSMC
- Channel Width = Variable

2.2 FPGA CAD Flow

The process of converting a circuit description into a format that can be loaded into an FPGA can be roughly divided into five discrete steps, namely: synthesis, technology mapping, clustering, placement and routing. The final output of FPGA CAD tools is a bitstream that configures the state of the memory bits in an FPGA. The state of these bits determine the logical function that the FPGA implements. Figure 2-3 shows a flowchart of the FPGA CAD flow. The following sections will describe the algorithms that are typically used in each step of the CAD flow.

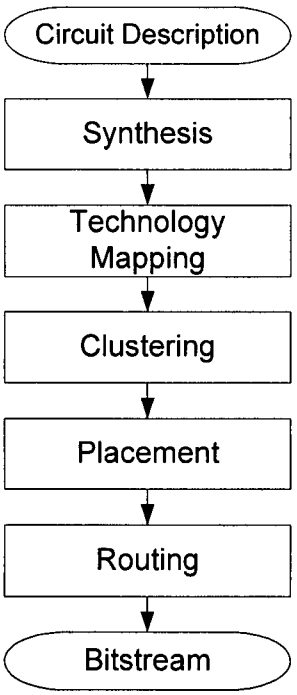

Figure 2-3: FPGA CAD Flow

2.2.1 Synthesis

Synthesis involves translating a circuit description, traditionally in a hardware description language (HDL) (e.g. VHDL or Verilog), into a gate-level representation. The gate-level representation is a network consisting of Boolean logic gates and flip-flops. There are no FPGA-specific optimizations performed during synthesis since this is normally a technology independent step. Further details concerning synthesis are omitted because it is beyond the scope of this thesis.

2.2.2 Technology Mapping

The output from synthesis tools is a circuit description of Boolean logic gates, flip-flops and the wiring connections between these elements. The circuit can also be represented by a directed acyclic graph (DAG). Each of the nodes in the graph represents a gate, flip-flop, primary input or primary output. Each of the wires in the graph represents the connections between the different circuit elements. Figure 2-4 shows an example of a DAG representation of a circuit.

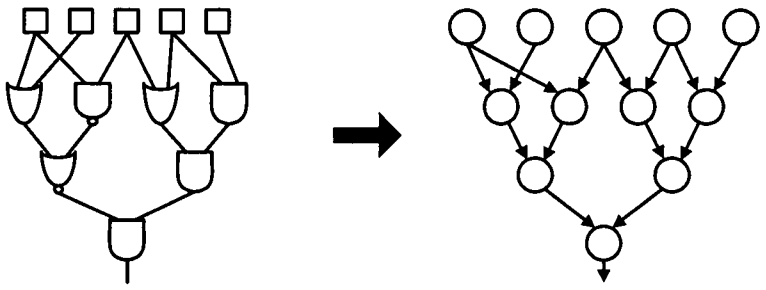

Figure 2-4: Directed Acyclic Graph Representation of a Circuit (from [29])

Given a library of "cells", the technology mapping problem can be expressed as finding a network of "cells" that implements the Boolean network. In the FPGA technology mapping problem, the library of "cells" is composed of K-input LUTs and flip-flops. Therefore, FPGA technology mapping involves transforming the Boolean network into K-bounded cells. Each cell can then be implemented as an independent K-LUT. Figure 2-5 shows an example of transforming a Boolean network into K-bounded cells.

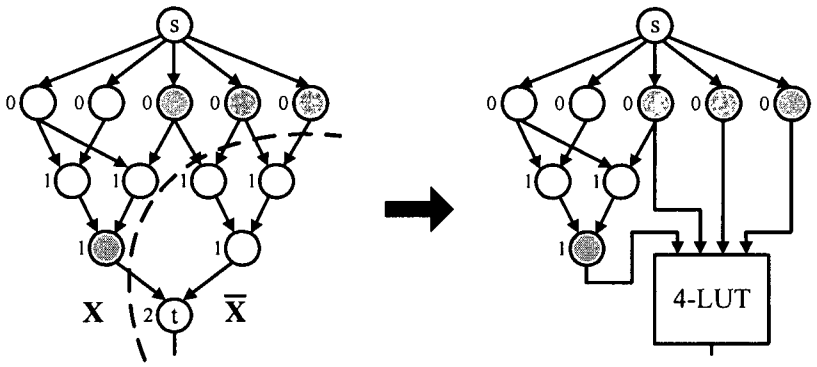

Figure 2-5: Example of Technology Mapping (from [29])

Technology mapping algorithms can optimize for a variety of objectives including depth, area or power. The FlowMap algorithm [12] is the most widely used academic tool for FPGA technology mapping. FlowMap was a breakthrough in FPGA technology mapping because it is able to find a depth-optimal solution in polynomial time. FlowMap guarantees depth optimality at the expense of logic duplication. Since the introduction of FlowMap, numerous technology mappers have been designed that optimize for area and run-time while still maintaining the depth-optimality of the

circuit ([13], [14], [15]). A series of technology mapping algorithms that optimize for power ([4], [11], [29]) have recently attracted much interest as well.

For the CAD tools discussed in this thesis, all technology mapping of circuits was performed by running FlowMap [12] for depth optimality and FlowPack [13] for area reduction. The SIS scripts *scipt.rugged* and *script.algebraic* were run and the lower area solution out of the two was chosen. The result of the technology mapping step generates a network of K-bounded LUTs and flip-flops.

2.2.3 Clustering

The logic elements in a mesh-based FPGA are typically arranged in two levels of hierarchy. The first level consists of basic logic elements (BLEs) which are K-input LUT and flip-flop pairs. The second level hierarchy groups 'N' BLEs together to form configurable logic blocks (CLBs). The clustering phase of the FPGA CAD flow is the process of forming groups of 'N' BLEs. These clusters can then be mapped directly to a logic element on an FPGA. Figure 2-6 shows an example of the clustering process.

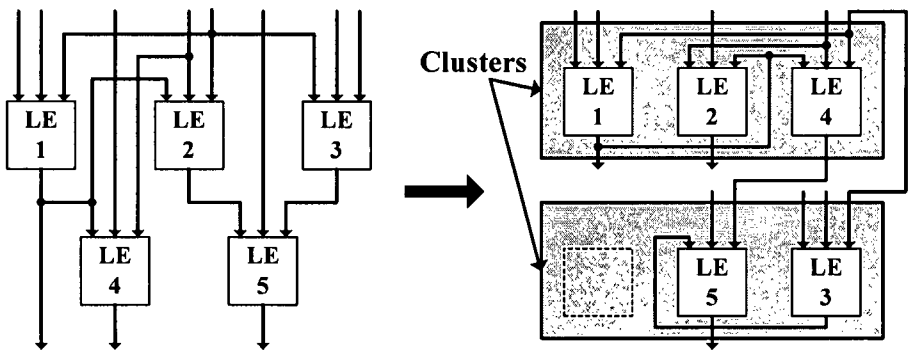

Figure 2-6: Example of Clustering (from [29])

Clustering algorithms can be broadly categorized into three general approaches, namely top-down ([20], [22]), depth-optimal ([16], [40], [54]) and bottom-up ([6], [7], [8], [37], [46]). Top-down approaches involves recursively partitioning a circuit into fixed size clusters. Depth-optimal solutions attempt to minimize delay (similar to [12]) at the expense of logic duplication. Bottom-up approaches are generally preferred for FPGA CAD tools because of their fast run times and reasonable timing delays.

Bottom-up approaches build clusters sequentially one at a time. The process starts with choosing a BLE which acts as a cluster seed. BLEs are then greedily selected and added to the cluster based on various attraction functions. The VPack [37] attraction function is based on the number of shared nets between a candidate BLE and the BLEs that are already in the cluster. T-VPack [6] is a timing driven version of VPack which gives added weight to grouping BLEs on the critical path together. RPack [7] improves the routability of a circuit by introducing a new set of routability metrics. RPack significantly reduced the required channel widths required by circuits compared to VPack. T-RPack [8] is a timing driven version of RPack which is similar to T-VPack by giving added weight to grouping BLEs on the critical path. iRAC [46] improves the routability of circuits even further by using an attraction function that attempts to encapsulate as many low fanout nets as possible within a cluster. If a net can be completely encapsulated within a cluster, there is no need to route that net in the external routing network. By encapsulating as many nets as

possible within clusters, routability is improved because there are less external nets to route in total.

For the experimental results discussed in this thesis, a replica of the iRAC algorithm was constructed based upon the description given in [46]. The replica was used because the original tool is no longer available. The replica implements the cluster seed and attraction function of the original iRAC algorithm but omits the Rent based input limiting function. Despite this, the iRAC replica achieves results within 2% of the number of external nets given in [46].

2.2.4 Placement

The placement step in the CAD flow involves placing the clustered netlist on to fixed locations on the FPGA. Figure 2-7 shows an example of the placement problem.

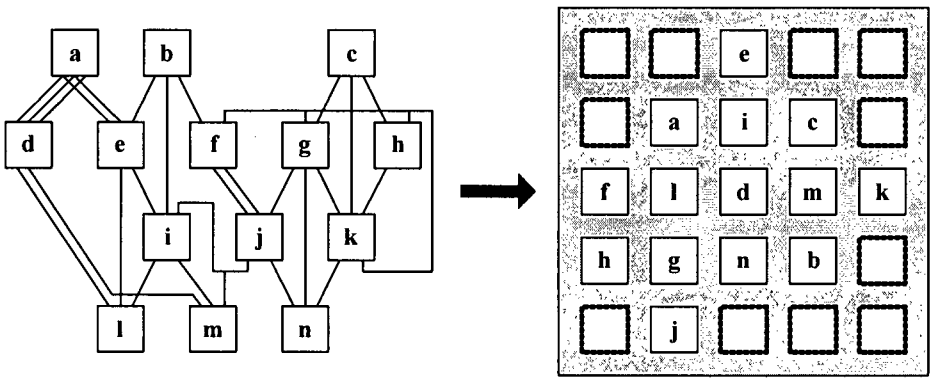

Figure 2-7: Example of Placement (from [29])

Placement algorithms traditionally attempt to minimize routing congestion and critical-path delays. Routing congestion is minimized by arranging the highly

connected blocks close together and critical-path delay is minimized by placing blocks on the critical path close together. Placement techniques can be broadly categorized into three different approaches: min-cut ([18], [24], [43]), analytical ([26], [42], [45]), and simulated annealing ([6], [27], [38]). Although all three methods produce good results, simulated annealing provides the most flexibility for new optimization goals and architectural changes.

Simulated annealing begins with a random initial placement of all the blocks. Pairs of blocks are then randomly swapped repeatedly. After each swap, the quality of the placement solution is analyzed. In VPR [6], the placement quality is determined by the sum of the half-perimeter bounding box of all the nets in the circuit. The probability of accepting a swap is based on the temperature of the simulated annealing schedule. Initially, the temperature is high which results in almost all swaps (good and bad) being accepted. As the temperature is slowly lowered, the probability of accepting a bad swap is reduced. Once the temperature reaches 0, only good swaps are permitted. The process of initially accepting bad swaps allows the placement process to find its way out of local minima in the solution space. For the CAD tools discussed in this thesis, the T-VPlace algorithm in the VPR tool is used unless otherwise specified.

2.2.5 Routing

The final stage in the FPGA CAD flow is the routing step which connects the placed blocks though the programmable routing network. Connections between wires on an FPGA are formed by using a programmable routing switch. Traditionally, wires

were bi-directional which indicates that tri-state drivers are placed on both ends of a wire. More recently, [33] has suggested that single driver, directional wires improve area and delay. However, since the directional VPR tool was unavailable at the time of this work, a bi-directional model for wiring was used.

Routing techniques can be broadly categorized into two methods, namely two-step routers ([10], [31], [32]) and combined global-detailed ([6], [39], [52]) routers. Two-step routers perform global routing and detailed routing in two discrete steps. Global routing assigns nets to specific channels and logic block pins. After global routing is complete, detailed routing assigns the nets to specific wire segments in its assigned channel. Two-step routers are generally used for ASIC flows and are not normally used for FPGAs because the limited flexibility of the FPGA routing architecture makes detailed routing difficult under global routing constraints. FPGAs use combined global-detailed routers because of the inflexibility of the two-step routers.

The VPR router (combined global-detailed) is based on a modified version of the PathFinder [39] algorithm. Pathfinder is an iterative algorithm which allows nets to share wires in the initial iterations. Successive iterations penalize the use of wires that were shared or used in previous iterations. The penalty factor is continually increased until a routing solution is found where each wire segment has at most one net assigned to it. The VPR router is also timing-driven. This is achieved by assigning the shortest possible paths to critical nets. Other non-timing critical nets may tend to take longer

routes in the presence of congestion. For the CAD tools discussed in this thesis, the T-VRoute algorithm in the VPR tool is used unless otherwise specified.

2.3 Previous Methods to Reduce Channel Width

One of the earliest attempts to balance logic and routing elements to decrease area was performed by DeHon [17]. However, this analysis was performed for an FPGA with a binary tree interconnect structure. In this work, we use a mesh based interconnect which is more representative of commercial FPGAs. Tessier [48] showed that depopulation of clusters can result in reduced MRCW of circuits. The algorithm presented in [48] depopulates each cluster equally so there is a uniform distribution of empty BLEs across the chip. Although this reduces MRCW, it also depopulates regions of the circuit that are not heavily congested. This leads to unnecessary CLB inflation in these regions. The tools presented in this thesis use a different cluster size limit for different partitions of the circuit. This cluster size limit value may vary across the chip such that routing-congested areas are depopulated more.

Singh [46] presented a clustering algorithm (iRAC) which is very effective at reducing channel width. iRAC reduces channel width by identifying low fan-out nets and completely absorbing them into a cluster. This reduces the total number of external nets, hence reducing the MRCW. iRAC also limits the number of inputs to each CLB by using the Rent parameter of the underlying architecture, resulting in solutions that have some depopulation. The tools in this thesis differ from [46] by targeting specific channel-width constraints.

Independence, a FPGA placement tool by Sharma [44], targets hard channel width and array size constraints. It works by using the router tool as an inner loop during placement and runs 10,000 times slower. In comparison, the tools presented in this thesis run much faster and can work with most clustering, placement and routing tools. Also, Independence inserts entire CLBs as whitespace, while the tools in this thesis insert individual BLEs.

Chapter 3

3 METHODS TO REDUCE MINIMUM ROUTABLE CHANNEL WIDTH

This chapter compares two basic techniques for channel width reduction. These methods are input-limiting and BLE-limiting. Results will show that contrary to popular belief (such as results in [46]), input-limiting is not effective at reducing channel width. Instead, BLE-limiting is shown to be much more effective.

3.1 Input-Limits vs. BLE-Limits

This section evaluates the effectiveness of two different CLB depopulation methods, namely input-limiting and BLE-limiting. The first method, similar to [48], is to strictly limit the number of BLEs that can be packed into a CLB (BLE-limit). The second method, similar to [46], is to strictly limit the number of inputs that can be used on a CLB (input-limit). Figure 3-1 shows the MRCW for circuit **clma** after implementing the two limits in two different clustering algorithms: (T-VPack, iRAC replica). Other circuits produce similar results. For example, a BLE-limit size of 7 would indicate that 9 of the BLEs in each cluster are empty. Alternatively, an input-limit size of 24 indicates that 27 of the inputs in every cluster are left unused.

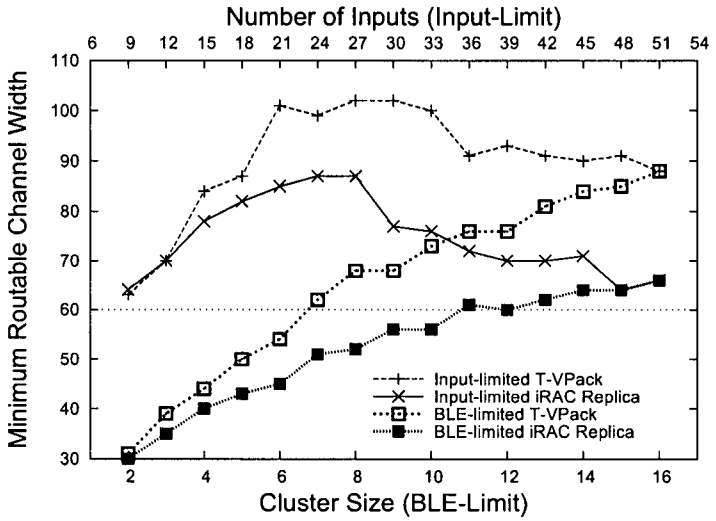

Figure 3-1: Input- and BLE-Limits during Clustering for Circuit clma

Figure 3-1 shows that the BLE-limit method exhibits a monotonically increasing relationship between the BLE-limit size and the MRCW. Hence, BLE-limit can be effectively used to decrease routed channel widths. Surprisingly, the input-limit approach did not exhibit this same relationship. This contradicts traditional thinking that reducing inputs is an effective way to reduce channel width. This occurs because there are two competing factors that affect the MRCW. As the BLE-limit or input-limit size is decreased, the increase in array size tends to **reduce** the MRCW but the increase in the total number of routable nets tends to **increase** the MRCW. BLE-limiting ensures that the array size increases more quickly than the number of routable nets, leading to a decrease in MRCW. In contrast, for the input-limiting case of Figure 3-1, as the number of used inputs decreases from 51 to 30, the number of routable nets is increasing while the array size remains relatively constant. Effectively, the reduction in the number of inputs is causing poor clustering solutions to be generated (e.g.

increase in total number of routable nets) without any increase in the required array size. Because BLE-limiting is an effective control method for reducing channel width, it is used as the depopulation method for all the depopulation tools discussed in this thesis.

Chapter 4

4 BENCHMARK CIRCUITS

Before the channel width reduction tools are presented, this chapter will discuss the importance of benchmark circuits to channel width reduction tools and present two new synthetic benchmark suites. FPGA researchers need large circuits to investigate new FPGA device architectures and CAD algorithms. However, the gap between the size of real world circuits and those available to the academic community for designing FPGAs continues to grow. Modern, multi-million gate System-on-Chip designs are highly proprietary; hence, they are not commonly available for academic research. Instead, the only designs available are small MCNC benchmark circuits [36] that have been in use since 1993. In an industry where circuit density doubles every 18-24 months, these circuits are rapidly becoming outdated.

A viable alternative to real world circuits is the use of synthetic circuits. Synthetic circuits can be generated using a variety of different methods. This chapter will present two different benchmark suites and discuss the mechanisms used to create the circuits. The first benchmark suite, Meta, was used to test the DHPack depopulation strategy described in Chapter 5. The second benchmark suite, Stdev, was used to test the Un/DoPack depopulation strategy described in Chapter 6.

4.1 Meta Benchmark Circuits

The System-on-Chip (SoC) design philosophy consists of integrating multiple components from different sources into a single chip. For FPGA systems, these components are normally digital intellectual property (IP) blocks. The IP blocks can be widely varied in their function and purpose, and are often developed by different designers. During development, each IP block might be individually placed and routed on an FPGA several times. As well, these different blocks may have different interconnect demands, just like those shown in Figure 1-1.

To mimic a large SoC design, the Meta circuit benchmarks were created by treating the largest 20 MCNC circuits [36] as individual IP blocks of a common SoC and *randomly stitching* them into a single, large Meta circuit. Stitching involves connecting compatible inputs and outputs of the blocks together. Each MCNC circuit is a unique, self-contained function with an appropriate input/output (I/O) count, just like an IP block. Connections between IP blocks are made only at these I/O boundaries and not to internal nodes of the block. Also, some of these MCNC circuits (*e.g.* **bigkey**) have many inputs and outputs, making them similar to "glue logic" that may be used to connect multiple IP blocks together. To avoid creating combinational loops, the stitching process adds a flip-flop to the primary outputs of each MCNC circuit. The IP blocks were stitched together in three different ways to create three different circuits in the Meta benchmark suite:

- **Independent:** Each primary input and primary output of each IP block remains a primary input and primary output of the Meta circuit. There is no interaction between IP blocks.
- **Pipeline:** The IP blocks are placed in a random, sequential order, each representing stages in a pipeline. Additional (leftover) inputs/outputs between pipeline stages become primary inputs/outputs of the Meta circuit.
- **Clique:** The outputs of each individual IP block are uniformly distributed to the inputs of all other circuits in the Meta circuit. The connections are made to encourage as much inter-block communication as possible.

When stitching, precise output-to-input connections are randomly assigned once. From this stitching assignment, multiple versions of each benchmark circuit were created by stitching different clustering solutions of each IP block. During stitching, only connections with a fan-out of one are formed. Alternatively, synthetic circuit generating techniques ([23], [28]) could have been used. These techniques are good for cloning existing circuits: they typically work by top-down partitioning or bottom-up clustering of modules and adding nets between the modules while enforcing stochastic interconnect parameters. Unfortunately, we do not have any initial SoC designs to clone. Another synthetic benchmark generator developed by Stroobandt [51] is discussed section 4.2.

When developing the Meta benchmarks, the primary concern was to create large circuits with varying interconnect usage among the IP blocks. The names of the

3 benchmark circuits are **Clique**, **Pipeline** and **Independent**. These benchmark circuits were used to test the DHPack depopulation strategy described in Chapter 5 which requires strict IP block boundary definitions.

4.2 Stdev Benchmark Circuits

The Meta benchmark suite was created by randomly stitching together existing, smaller benchmarks (MCNC circuits) and treating the smaller circuits as IP blocks. However, the stitching was somewhat unrealistic as a flip-flop was placed at every IP block output to prevent combinational loops. Un/DoPack (the depopulation technique described in Chapter 6) does not have the requirement that the circuit be strictly partitioned into IP blocks. To mimic more realistic benchmark circuits, a synthetic benchmark generator, GNL [51] was used to generate a second benchmark suite. GNL allows benchmarks to be generated hierarchically and allows control over the Rent parameter [30] in each division. GNL is also able to prevent combinational loops and can place limits on the maximum depth of a circuit. The key parameter of GNL is that it is able to create synthetic benchmarks based on Rent's rule. Empirical evidence has shown that most circuits follow Rent's rule. Since it is not known how much interconnect variation is present in real world circuits, GNL gives provides a mechanism to generate circuits that have a controllable amount of interconnect variation.

The GNL synthetic circuits generated consist of two levels of hierarchy. The root level defines the overall structure of the circuit. This level includes the total

number of logic cells in the circuit, as well as a required input and output count. The number of primary inputs and outputs were defined as 240 and 120 respectively. The root level is defined such that it is made up of twenty leafs that mimic the 20 largest MCNC circuits [36]. Each leaf represents an IP block with a specific Rent parameter. The Rent parameter and number of logic blocks of each IP block was chosen to match the same parameter values as each corresponding MCNC circuit. These Rent numbers were extracted from [46]. The number of inputs and outputs for each sub-circuit was not defined, thus allowing GNL to randomly stitch each Rent region together to form the overall circuit. The standard deviation of the Rent parameter for the 20 MCNC circuits was calculated to be 0.08 and the average value was 0.62. Using these Rent values, we produced a ***clone*** of the Meta circuit and named it Stdev008. To create a family of circuits, a linear interpolation scheme was applied to keep the same overall mean, but to vary the standard deviation to produce 4 smaller values and 2 larger ones. Figure 4-1 shows a graphical representation of our linear interpolation scheme. For clarity, only 10 of the 20 MCNC circuits are shown.

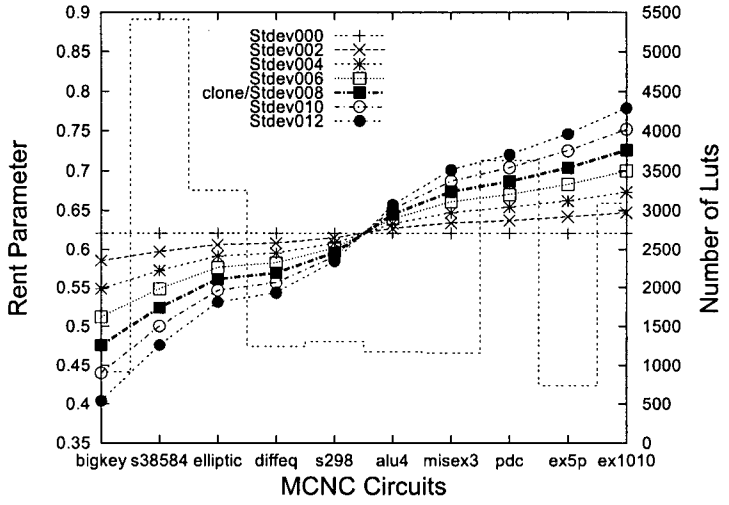

Figure 4-1: Rent Linear Interpolation for GNL Benchmark Circuits

Each line in Figure 4-1 represents a benchmark circuit for a specific standard deviation of the Rent parameters. Circuit Stdev000 contains 20 IP blocks each having the identical Rent parameter of 0.62, producing a flat line. The average Rent parameter is a simple average of the sub-circuits and is not weighted by the sub-circuit size. Three other circuits with standard deviations 0.02, 0.04, 0.06 were created between the flat line and bold **clone** circuit line. Circuits Stdev010 and Stdev012 were obtained by extrapolating the Rent parameter 2 steps farther. The "bar line" in Figure 4-1 shows the size of each of the IP blocks in terms of the number of LUTs; the size does not depend on the Rent parameter.

The resulting circuits had standard deviations of 0.0, 0.02, 0.04, 0.06, 0.08, 0.10 and 0.12 in their Rent value and contained 51,900 to 52,200 6-input BLEs. The names of the circuits are **Stdev000, Stdev002, Stdev004, Stdev006, Stdev008/clone, Stdev010** and **Stdev012**. The linear interpolation scheme will

be used to show that a large amount of depopulation is necessary to reduce the MRCW in circuits with a low standard deviation. This is because the circuit is uniform, and routing resources demands should be fairly consistent across the entire circuit. In contrast, with a high standard deviation, routing resource demands should be non-uniform, thus allowing the depopulation scheme to reduce the routing demands of high Rent regions. Appendix A gives complete information on the size and Rent parameter of the IP blocks in the Stdev benchmark circuits.

It was not possible to use the Stdev circuits to test DHPack (Chapter 5) because DHPack requires that the IP block boundaries be strictly defined. Even though it is possible to specify the Rent parameter of each IP block, GNL is still random in nature. Therefore, there is no method of determining where the boundaries of these IP blocks are located. The Stdev benchmarks will be used primarily in Chapter 6 to demonstrate the significance of interconnect variation in channel width reduction strategies. Experimental results will show that Un/DoPack is effective at reducing MRCW for the Stdev circuits and the Meta circuits regardless of their interconnect variation. However, the amount of interconnect variation has a direct affect on the overall area increase and run time of this tool.

Chapter 5

5 CHANNEL WIDTH REDUCTION USING DESIGN HIERARCHY PACKING: DHPACK

This chapter describes a non-uniform depopulation technique (DHPack) that uses the natural design hierarchy of the benchmark circuits to identify depopulation regions. DHPack requires that the benchmark circuits have clearly defined IP block boundaries. Since the Meta benchmarks (Section 4.1) were created from a strict design hierarchy, they were ideal for evaluating DHPack. The Meta circuits were formed by stitching together individual clustering solutions of each IP block. This allows DHPack to depopulate only the routing-intensive blocks. Clustering individually preserves each IP block in a form that more closely resembles how each was developed and tested by separate designers prior to integration.

This chapter begins with an explanation of the DHPack algorithm. An analysis of the experimental results will then show that this technique is effective at reducing the minimum routable channel width (MRCW) of the benchmark circuits. The chapter concludes with a discussion of some of the limitations of this technique. Note that this technique is described in [49].

5.1 DHPack - Depopulation Strategy

Design Hierarchy Pack (DHPack) uses the design hierarchy of the benchmark circuit to identify depopulation regions. This approach enforces BLE-limits during clustering, profiles each IP block's channel width needs for different depopulation levels, and chooses the one with the fewest CLBs that meet a given channel width constraint. Results will show that a large, flat area region exists where CLB count can be safely traded off for channel width.

For FPGA designs that contain multiple IP blocks, it was hypothesized that the channel width needed to route the entire circuit will be similar to the IP block with the highest channel width needs. That is, the other IP blocks do not temper the channel width needs of the hard-to-route IP block. Although this is just a first-order approximation that ignores the effects of inter-block communication, results show that it is a good estimate of the final routed channel width. Hence, the first step of DHPack is to develop a channel-width profile of each IP block. Then, DHPack selects the depopulation level needed by each IP block to meet the overall channel-width constraint. DHPack is described in pseudo-code in Figure 5-1. Each of the 4 different steps of DHPack is discussed in the following sections.

```
Routed_Circuit DHPack ( circuit, channel_width_constraint,
                                          cluster_size ) {

      IP_Blocks[ ] = Decompose_Circuit_into_IP_Blocks( circuit );

      // Step 1: Generate Channel Width Profile
      foreach (IP_Block) {
            for(LimitSize=1; LimitSize<=cluster_size; LimitSize++) {
                  cluster_ip_blk = Cluster( IP_Block, LimitSize );
                  routed_ip_blk = Place&Route( cluster_ip_blk );
                  CW[IP_Block][LimitSize] = get_CW( routed_ip_blk );
            }
      }

      // Step 2: Calculate Maximal Cluster Sizes
      foreach (IP_Block) {
            Limit = Cluster_Size;
            while( CW[IP_Block][Limit] > channel_width_constraint &&
                   Limit > 0) {
                  Limit--;
            }
            if( Limit == 0 ) {
                  return( NO_SOLN );
            } else {
                  BLE_Limit[IP_Block] = Limit;
            }
      }

      // Step 3: Cluster IP Blocks
      foreach (IP_Block) {
            Clustered_Soln[IP_Block] = Cluster( IP_Block,
                                          BLE_Limit[IP_Block]);
      }

      // Step 4: Stitch Circuit back together & P&R
      Clustered_Circuit = Stitch_Circuit(circuit, Clustered_Soln[ ]);
      Routed_Circuit = Place&Route( Clustered_Circuit );

      return ( Routed_Circuit )

}
```

Figure 5-1: Pseudo-code for DHPack Flow

5.1.1 Steps 1,2: Channel Width Profiling and BLE-Limits

The channel width profile of each IP block in the Meta circuits were created by placing and routing each IP block independently of each other for all possible BLE-limit sizes. Figure 5-2 shows the channel width needs of two IP blocks for BLE-limits 2 to 16. A BLE-limit size of 16 indicates that the clustering tool has no restriction on the number of BLEs that can be used in a cluster. Conversely, a BLE-limit size of 2 indicates that a maximum of 2 BLEs can be used per cluster.

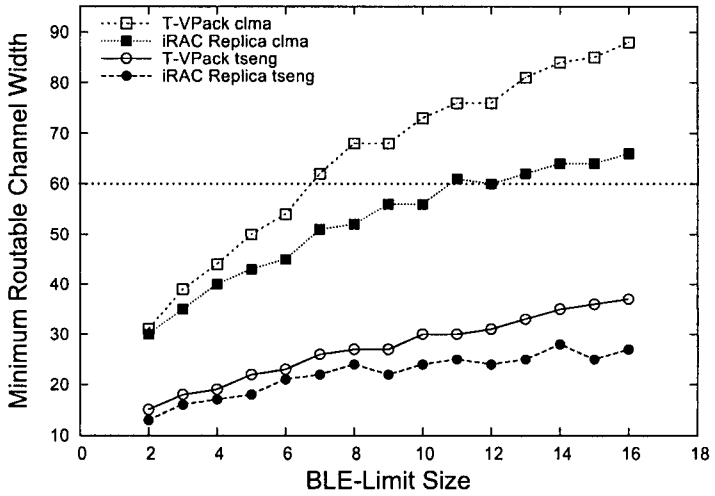

Figure 5-2: Channel Width Profile of IP Blocks clma/tseng

If a channel-width constraint of 60 is imposed using T-VPack, a BLE-limit size of 6 is required to route clma. We say 6 is the **maximal cluster size** for clma at the given channel-width constraint. In contrast, the maximal cluster size for tseng is 16 for the same constraint. Once a channel width profile is created for each IP block in the design, the maximal cluster size for each IP block can be calculated given a channel width constraint.

For the 3 Meta circuits, 11 different channel-width constraints were set and the maximal cluster sizes were determined for each IP block using both T-VPack and the iRAC replica. The maximal cluster sizes using T-VPack are shown in Table 5-1. Channel-width constraints below 45 were not possible because some circuits could not be depopulated enough to route with such a small channel width. Channel-width constraints greater than 95 were not interesting because all CLBs were fully populated. A table with the maximal cluster sizes using the iRAC replica is also given in Table 5-2.

	Channel-Width Constraint										
Circuit	**95**	**90**	**85**	**80**	**75**	**70**	**65**	**60**	**55**	**50**	**45**
alu4	16	16	16	16	16	16	16	16	16	16	16
apex2	16	16	16	16	16	16	16	16	16	16	12
apex4	16	16	16	16	16	16	16	14	10	9	8
bigkey	16	16	16	16	16	16	16	16	16	14	9
clma	16	15	14	12	11	10	8	6	5	5	3
des	16	16	16	16	16	16	16	15	4	3	2
diffeq	16	16	16	16	16	16	16	16	16	16	16
dsip	16	16	16	16	16	16	16	16	16	13	6
elliptic	16	16	16	16	16	16	16	14	11	9	7
ex1010	16	16	16	16	16	15	12	9	7	5	4
ex5p	16	16	16	16	16	16	16	16	16	16	15
frisc	16	16	16	15	13	10	9	7	7	5	4
misex3	16	16	16	16	16	16	16	16	16	16	16
pdc	16	16	16	16	16	14	12	9	7	6	4
s298	16	16	16	16	16	16	16	16	16	16	16
s38417	16	16	16	16	16	16	16	16	16	16	14
s38584	16	16	16	16	16	16	16	16	13	11	9
seq	16	16	16	16	16	16	16	16	16	15	11
spla	16	16	16	16	16	16	13	11	8	6	5
tseng	16	16	16	16	16	16	16	16	16	16	16

Table 5-1: Maximal BLE-Limit Sizes from T-VPack

Circuit	Channel-Width Constraint										
	80	76	72	68	64	60	56	52	48	44	40
alu4	16	16	16	16	16	16	16	16	16	16	16
apex2	16	16	16	16	16	16	16	16	16	14	9
apex4	16	16	16	16	16	16	16	13	11	8	5
bigkey	16	16	16	16	16	16	15	15	15	15	12
clma	16	16	16	16	13	10	8	7	6	5	3
des	16	16	16	16	16	16	15	15	15	15	15
diffeq	16	16	16	16	16	16	16	16	14	12	8
dsip	16	16	16	16	16	16	16	15	15	15	15
elliptic	16	15	12	11	9	8	6	5	5	4	3
ex1010	16	16	16	16	15	11	8	8	5	4	3
ex5p	16	16	16	16	16	16	16	16	16	16	11
frisc	16	16	16	16	14	10	9	8	8	6	4
misex3	16	16	16	16	16	16	16	16	16	16	16
pdc	16	16	16	16	12	10	9	6	5	5	3
s298	16	16	16	16	16	16	16	16	16	16	16
s38417	16	16	16	16	16	16	16	16	16	16	16
s38584	16	16	16	16	16	16	16	16	16	16	15
seq	16	16	16	16	16	16	16	16	16	12	8
spla	16	16	16	16	16	14	11	9	7	5	4
tseng	16	16	16	16	16	16	16	16	16	16	16

Table 5-2: Maximal BLE-Limit Sizes from iRAC

5.1.2 Steps 3,4: Cluster IP blocks and Stitch Circuit

Once the maximal cluster sizes have been determined for a given channel width constraint, DHPack selects the individual clustering solutions for each IP block and stitches the circuit back together. When clustering the IP blocks, there are two choices for the BLE-limit size with a given channel-width constraint:

- **Uniform (Minimum) Cluster Size:** Depopulate all of the IP blocks to the same BLE-limit size, the minimum of the maximal cluster sizes for all IP blocks. This is similar to [48] which uses uniform depopulation of clusters.

- **Non-uniform (Maximal) Cluster Size:** Depopulate the IP blocks by different amounts, using the maximal cluster size for each one.

For each of the 11 channel width constraints in Table 5-1, we generated a **Uniform** and **Non-uniform** clustered version of Meta using T-VPack. This was also repeated for the iRAC replica algorithm. As discussed earlier, the **Uniform** version will contain more CLBs than necessary and results in lower BLE utilization. Figure 5-3 shows the total CLBs and BLE utilization obtained from the Meta circuits produced from T-VPack and iRAC replica clustering.

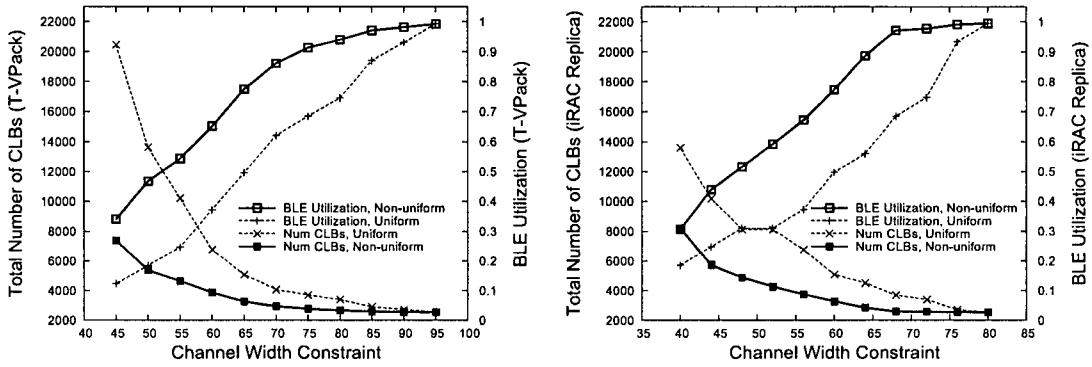

Figure 5-3: DHPack CLB Count and BLE Utilization

It is evident from Figure 5-3 that **Non-uniform** clustering of the IP blocks significantly improves both BLE utilization and reduces CLB count as the channel width constraint is decreased

5.2 Experimental Results

In total, 66 Meta netlists were created and placed using VPR (11 channel width constraints, 3 Meta circuits, using 2 clustering tools (T-VPack and the iRAC replica)).

Figure 5-4 shows a post place and route screen shot from VPR of the Meta circuit Clique with a channel width constraint of 50. The screen shot has been edited to show the location of the IP blocks.

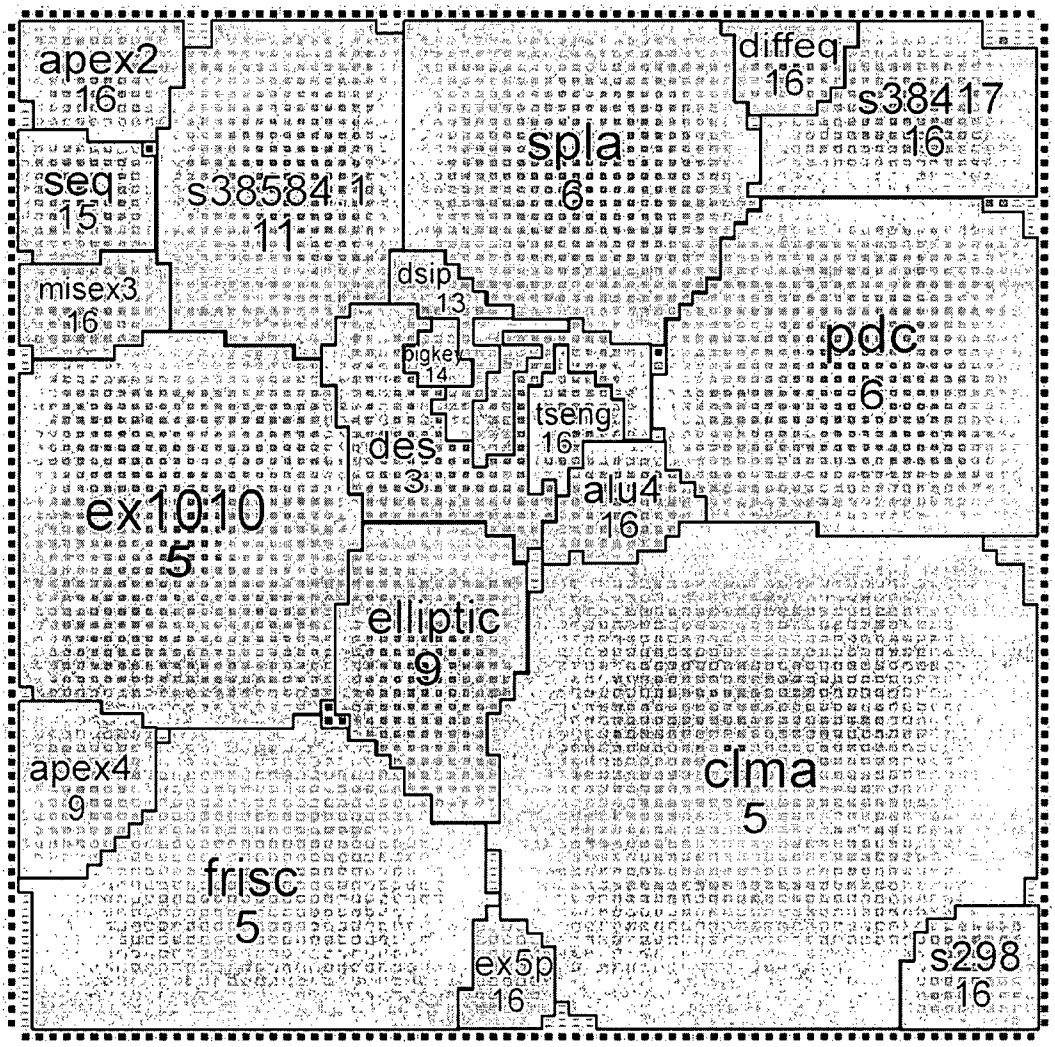

Figure 5-4: VPR Placement of Non-Uniform Clique with T-VPack

The numbers below each IP block indicate the BLE-limit size of that IP block. Analyzing the place and route results from the Meta circuits led to two key observations.

Observation 1: VPR placer successfully groups IP blocks from a random initial placement

It was expected that large SoC designs will be floor planned prior to the final placement process, but VPR does not support floor planning. Instead, it starts with a random placement of all CLBs and uses simulated-annealing to find a minimum-cost placement. Interestingly, VPR was able to generate solutions that appear to be floor planned. This reduced the need to impose an artificial floor plan on the design a priori.

Observation 2: VPR router confirms the MRCW of a Meta circuit is dominated by a few IP blocks

Figure 5-4 illustrates that only a few IP blocks (i.e. des, clma, frisc, ex1010) needed a large amount of depopulation for the given channel width constraint. It is these IP blocks that dominate the channel width needs of the entire circuit.

While routing, the channel width was continuously reduced until the circuit became un-routable. This produced the final minimum routable channel width (MRCW). It is a minimum because the FPGA architecture must have at least this minimum channel width in order for the circuit to be routable. Routing results for the 6 Meta circuits are shown in Figure 5-5. A comprehensive table of results is given in Appendix B.

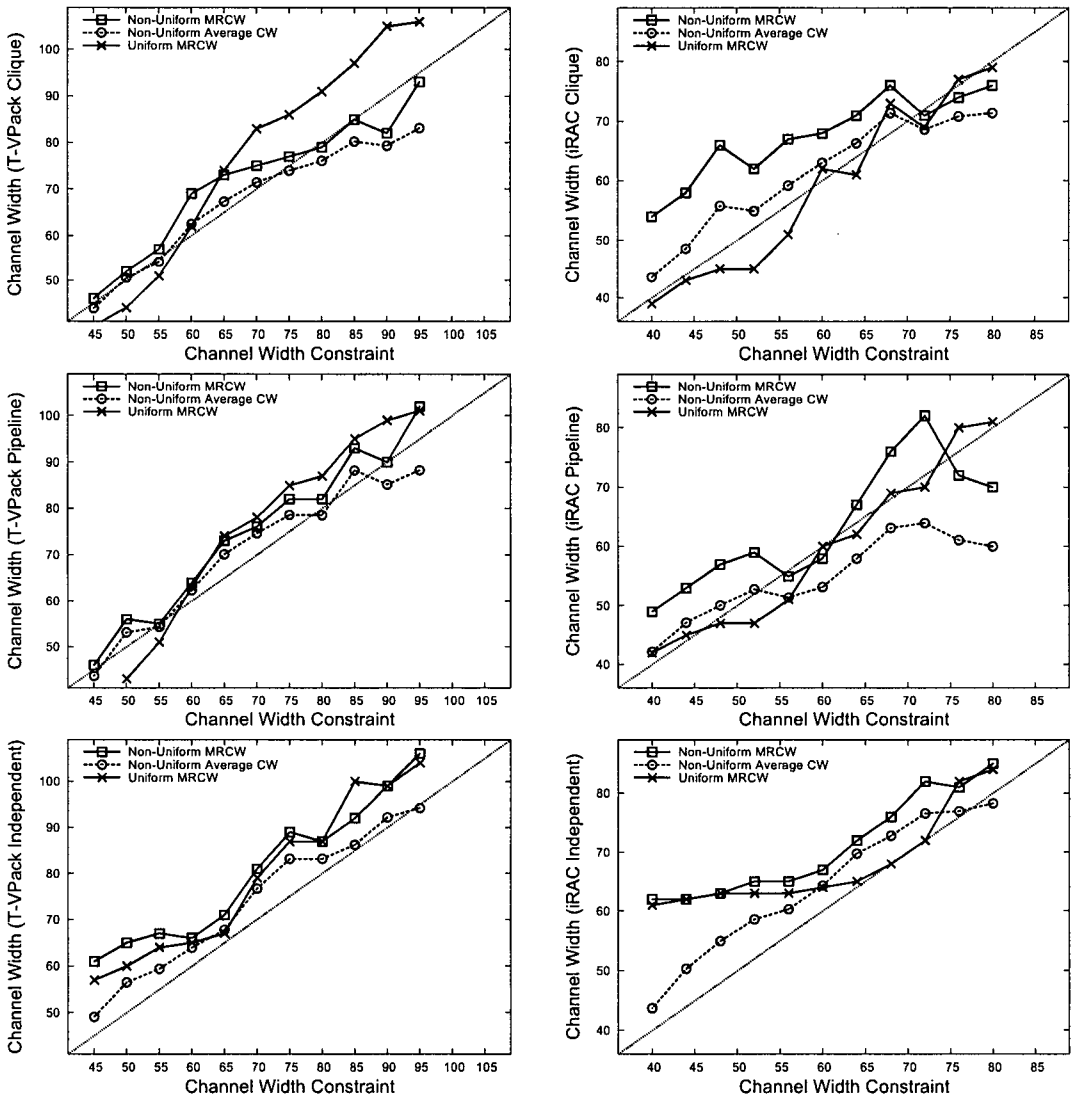

Figure 5-5: DHPack MRCW and Average Channel Width

The results show that the non-uniform MRCW was usually higher than what was imposed by the channel-width constraint. In contrast, the uniform MRCW results track the channel width constraint more closely. However, this comes at the expense of area which is shown in Figure 5-6. The high MRCW values for some Independent and Pipeline cases involving iRAC led to further investigation. It was found that on

these occasions, the I/O intensive IP blocks were strongly attracted to the I/O padframe during placement and stretched into highly rectangular shapes. This caused severe localized congestion in the routing channels nearest to the padframe. Figure 5-5 also shows the average channel width of all routing channels. The average channel width tracks the channel-width constraint much more closely, suggesting that the approach is viable if the I/O padframe congestion can be reduced.

Figure 5-6 shows the normalized area results of the 3 Meta circuits for non-uniform and uniform depopulation cases. For the non-uniform case, the final routed area shows a substantially flat area response (small area increases) for channel widths of 70-95 for T-VPack and 65-80 for iRAC. Channel width decreases of up to 50% are possible with much larger area increases. In comparison, the uniform area curves increases much more quickly as the channel width constraint is decreased. This suggests that uniform depopulation unnecessarily depopulates in low-congestion areas.

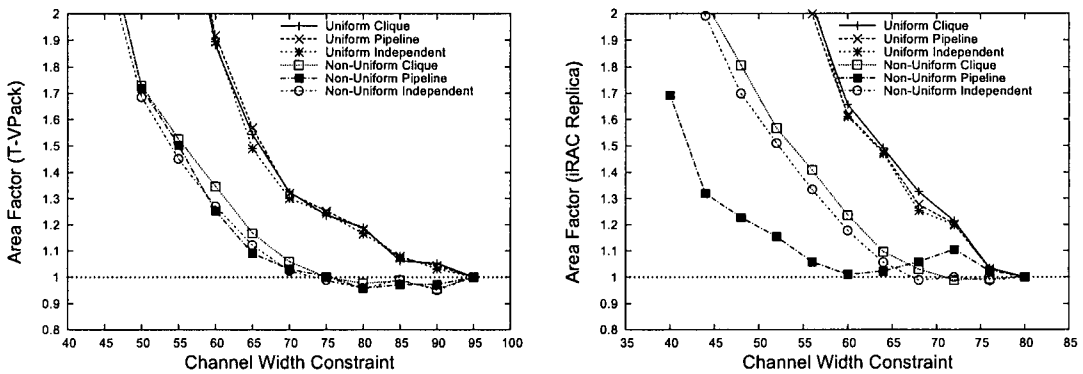

Figure 5-6: DHPack Routed Area Factor

Table 5-3 shows a summary of the channel width decreases that were obtained for each Meta benchmark circuit. Small MRCW decreases of 23%/13% for T-

VPack/iRAC are possible with 4%/3% increase in area. Larger MRCW decreases of 39%/29% are possible with 166%/146% increase in area.

Circuit	Clustering Tool	Channel Width Changes					
		CW	Avg CW	Area	CW	Avg CW	Area
Clique	T-VPack	-19%	-14%	+6%	-50%	-47%	+129%
	iRAC Rep.	-7%	-4%	-1%	-29%	-39%	+187%
Pipeline	T-VPack	-25%	-15%	+2%	-55%	-51%	+184%
	iRAC Rep.	-17%	-11%	+1%	-30%	-30%	+69%
Independent	T-VPack	-24%	-19%	+3%	-42%	-48%	+184%
	iRAC Rep.	-15%	-11%	+6%	-27%	-44%	+183%
Arithmetic Mean	T-VPack	**-23%**	**-16%**	**+4%**	**-49%**	**-49%**	**+166%**
	iRAC Rep.	**-13%**	**-9%**	**+3%**	**-29%**	**-38%**	**+146%**

Table 5-3: Reductions in Channel Width for DHPack

In some cases, only small decreases in MRCW were achievable. As explained earlier, this is because some IP blocks introduce heavy congestion at the periphery due to high I/O padframe needs. Table 5-3 also shows the average channel width required. In the cases where I/O congestion occurs, the average channel width tracks the channel width constraint more closely. This suggests that if the I/O congestion can be somehow eliminated the MRCW will also decrease and track the channel width constraint more closely.

Figure 5-7 shows the critical-path delay results. It was initially expected critical-path delay would increase as more depopulation is applied. Critical-path delay does seem to follow this trend, but it tends to "jump around". This delay "noise" appears to result from instability in the placement. As depopulation is applied, the VPR placement engine keeps IP blocks together, but sometimes their location in the floor plan is shifted significantly relative to other IP blocks. This caused the critical

path to sometimes relocate from within an IP block (which gradually degrades as depopulation is applied) to connections between IP blocks (which introduces large delay jumps). Imposing a pre-defined floor plan may help reduce this "noise" in large designs.

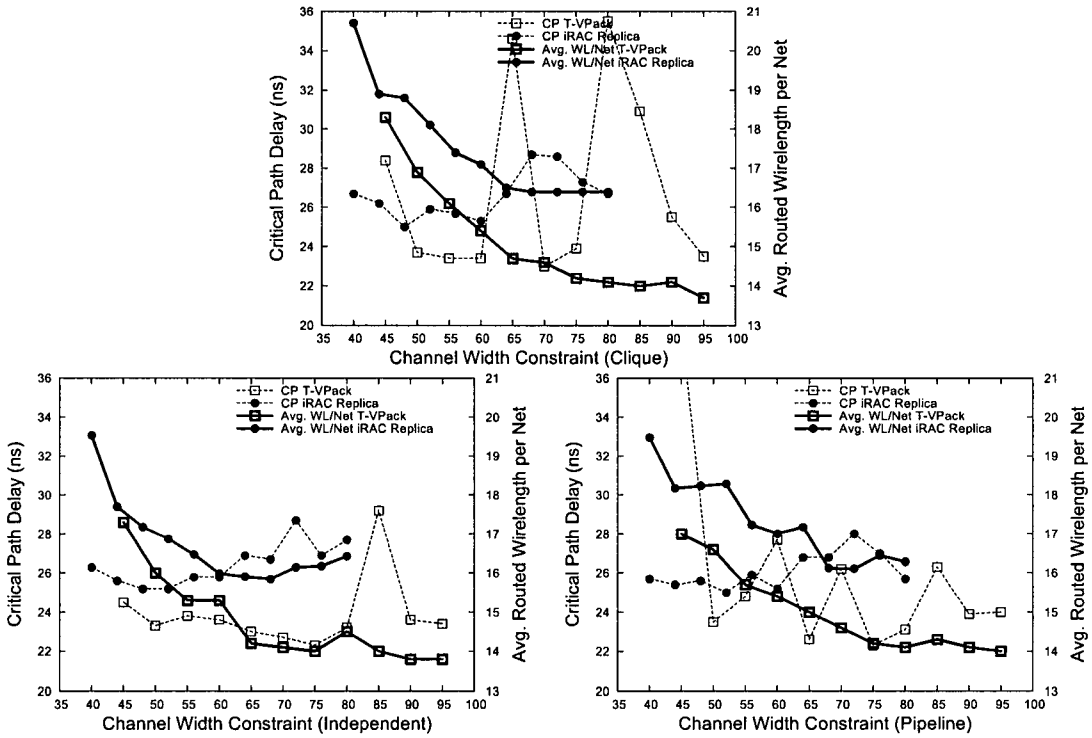

Figure 5-7: DHPack Critical-Path Delay

Figure 5-7 also shows the average wirelength per net. Average wirelength per net increased as more depopulation is applied (e.g. channel width constraint reduced). This is expected because an increase in CLB count must also increase the average distance a net must traverse. Also, depopulating will cause connections that were previously internal to a CLB (hence, ignored) to become external nets with a small measurable distance. This slightly tempers the increase in average wirelength. Note

that iRAC replica produces a higher average wirelength than T-VPack. However, the total wirelength was lower and the critical-path delay results were similar.

5.3 Experimental Conclusions

This chapter has proposed a system-level technique for mapping large system-on-chip (SoC) designs to channel-width constrained FPGAs. In particular, the method helps fit hard-to-route circuits into FPGAs that have narrow channel widths at the expense of using more CLBs. Since larger devices with more CLBs are usually available, this is a practical trade-off.

Results have shown that depopulating CLBs (e.g. not filling them to capacity) is a very effective way to reduce channel width needs of a circuit. It is important to apply non-uniform depopulation when clustering. Otherwise, area increases very rapidly and limits the usefulness of the approach. It was shown that channel width reduction can be achieved by selectively depopulating parts of a large circuit that would otherwise have routing congestion. The most routing-intensive IP blocks are depopulated until the routing demands of those blocks are comparable to the demands of the other blocks. On average, small MRCW decreases of 23%/13% for T-VPack/iRAC are possible with 4%/3% increases in area. Large MRCW decreases of 39%/29% are possible with 166%/146% increases in area. Although this is a high area cost, it may be the only viable solution in a real FPGA device where hard channel-width constraints are imposed. By purchasing an FPGA device with higher logic capacity, designs which are otherwise un-routable can be made routable.

5.4 Technique Limitations and Future Work

This section will discuss some of the limitations of DHPack and some possible directions for future work.

5.4.1 I/O Padframe Congestion

The main reason why DHPack was not able to track the channel width constraint for large channel width decreases is because some IP blocks stretched into highly rectangular shapes along the I/O padframe causing congestion hotspots along the channel adjacent to the I/Os. Xilinx FPGAs have added additional routing resources to the I/O channel that runs in between the I/O pads and the logic blocks so that I/O pad placement does not impact routability and speed [47]. Hallschmid [21] also investigated the impact the aspect ratio of a circuit has on the required channel width. [21] suggests that a square aspect ratio generates the lowest channel widths and that larger aspect ratios increase the required channel width because the majority of the signals run along the tracks in the longer dimension. Table 5-3 demonstrated that the *average* channel width tracked the channel width constraint more closely than the MRCW. This suggests that if the channels adjacent to the I/O padframe were larger relative to the rest of the chip, channel width reductions could be improved. Another alternative is to tune the placement algorithms to avoid creating this congestion at all. These techniques need further investigation.

5.4.2 IP Block Granularity Too Coarse

DHPack is dependent on the design having a well-defined IP block partitioning. Often, SoC designs have multiple levels of hierarchy which makes it difficult to choose appropriate boundaries. DHPack does not allow the exploration of other design partitions. A more efficient partitioning of the circuit that does not rely on the design hierarchy may more accurately identify high congestion regions. This issue is addressed in Chapter 6 which presents a CAD tool (Un/DoPack) that does not rely on design hierarchy information.

5.4.3 Hard Channel Width Constraints

Even though a channel width constraint is an input parameter, DHPack may generate a routed solution that exceeds the constraint. In practice, industrial FPGAs have hard channel width constraints and routed solutions that exceed the constraint by even 1 track are not routable. Un/DoPack in Chapter 6 addresses this limitation by iterating to meet hard channel width constraints.

5.4.4 Congestion Profile Run Time Long

The run time to create the congestion profile of each IP block can be very time consuming. In our Meta circuit example, each of the 20 IP blocks needed to be placed and routed individually for BLE-limit size 2 to 16 (15*20=300 place and route executions). It may be argued that since each IP block is assigned to a different engineer that this channel width profiling must be done before integration into the

overall system. Nonetheless, this profiling step will increase the total CAD time significantly.

Chapter 6

6 CHANNEL WIDTH REDUCTION USING AUTOMATED CONGESTION IDENTIFICATION: UN/DOPACK

This chapter describes a depopulation technique (Un/DoPack) that iteratively applies non-uniform depopulation on a circuit until a given channel width constraint is met. The main difference between Un/DoPack and DHPack is that Un/DoPack is a multi-pass technique whereas DHPack is single-pass. Un/DoPack does not have the requirement that the design hierarchy be known a priori. It can be applied on any circuit irrespective of whether the design hierarchy is known or not.

This chapter begins by describing the Un/DoPack algorithm including a detailed discussion of each step of the CAD flow. The experimental results will show that Un/DoPack is effective at reducing channel width. The results will also highlight the importance of interconnect variation in benchmark circuits for determining the device channel width in FPGA architecture design. Finally, a comparison between DHPack and Un/DoPack is made followed by some of the limitations of this technique.

6.1 Un/DoPack - Depopulation Strategy

Figure 6-1 shows a flowchart of the Un/DoPack algorithm.

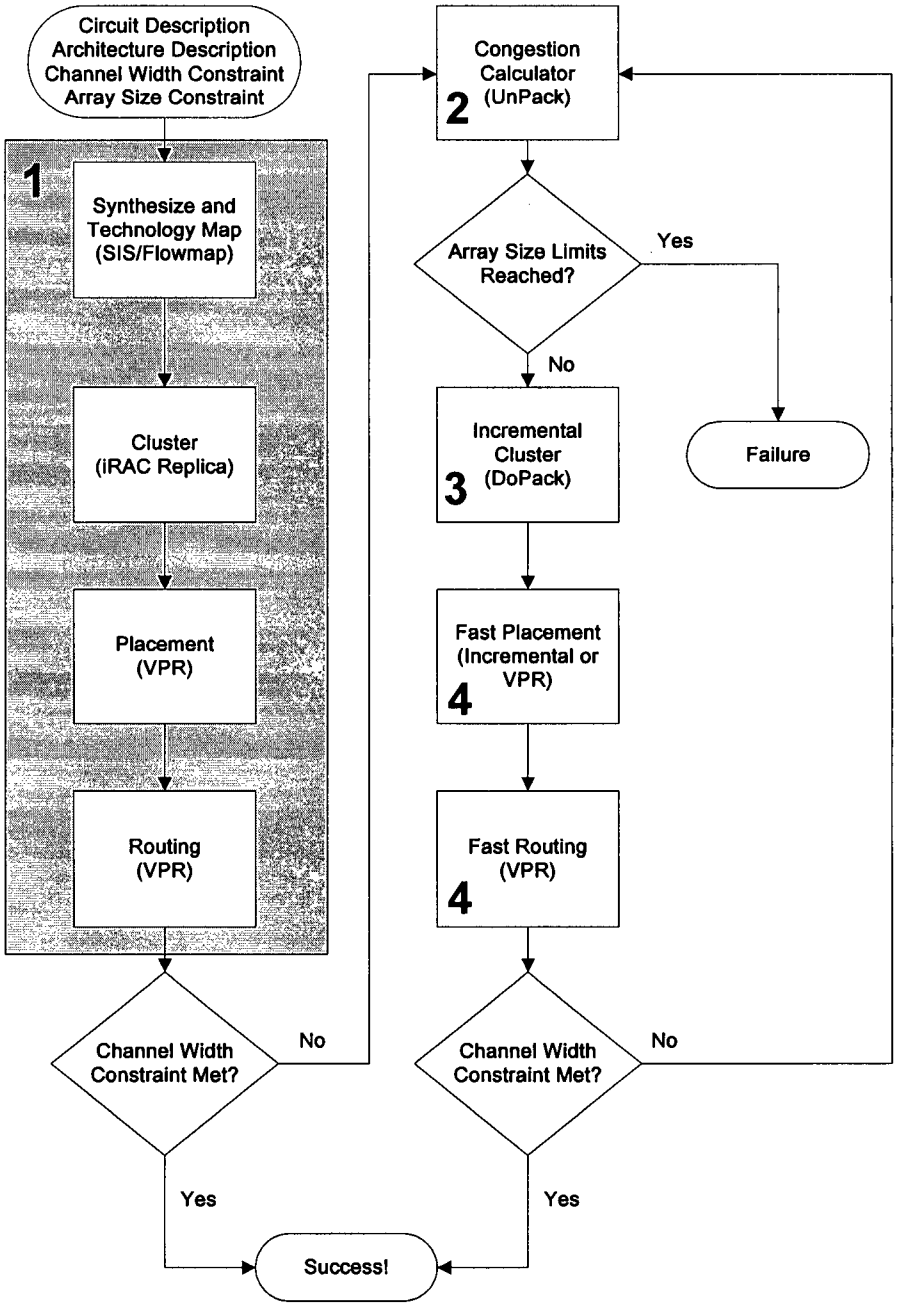

Figure 6-1: Un/DoPack CAD Flow

Un/DoPack can be roughly divided into 4 steps. Each of these steps are highlighted in Figure 6-1 and discussed below.

6.1.1 Step 1: Traditional SIS/VPR Flow

The first step is highlighted in the shaded portion of Figure 6-1. This step is the traditional academic FPGA CAD flow which uses SIS / FlowMap [12] and VPR [6] to synthesize, cluster, place and route a circuit. If the traditional CAD flow fails to produce a routed solution given a fixed channel width constraint, the iterative portion of Un/DoPack is invoked to reduce the MRCW.

There are four inputs to Un/DoPack: the circuit description, the architecture description, the channel width constraint and the maximum array size (logic capacity constraint). Traditionally, VPR performs a binary search on the device channel width until the MRCW is found. Un/DoPack does not perform this binary search but requires the user to specify a hard channel width constraint. This has many practical applications since industrial FPGAs have hard channel width constraints as well. The iterative portion of Un/DoPack is invoked only if the routed solution does not meet the given channel width constraint and the logic capacity of the FPGA will not be exceeded through depopulation.

6.1.2 Step 2: UnPack - Congestion Calculator

The second step (UnPack) determines which portion of the circuit to depopulate, calculates the amount of depopulation required as a new cluster size

constraint (BLE-limit size), and un-packs the BLEs. A smaller BLE-limit size constraint ensures the new CLBs will be "less full" than before.

Following a failed routing attempt, UnPack creates a congestion map based on the final routed solution. The congestion map is created by *labeling* each CLB with the maximum of the required channel width in each of the four channel segments adjacent to the CLB. Some wiring tracks may have multiple nets assigned to it from the failed routing solution. This is acceptable as the required channel width is estimated by counting the total number of nets **traveling** through the channel segment.

Figure 6-2 shows a sample 3-D congestion map of circuit Stdev008 before (top) and after (bottom) Un/DoPack meets a channel width constraint of 100. The x-y coordinates indicate the CLB locations and the z coordinate indicates the CLB congestion label. The peak / avg / stddev of channel utilization were 120 / 79.4 / 26.9 tracks before Un/DoPack, and 100 / 79.2 / 19.6 afterwards.

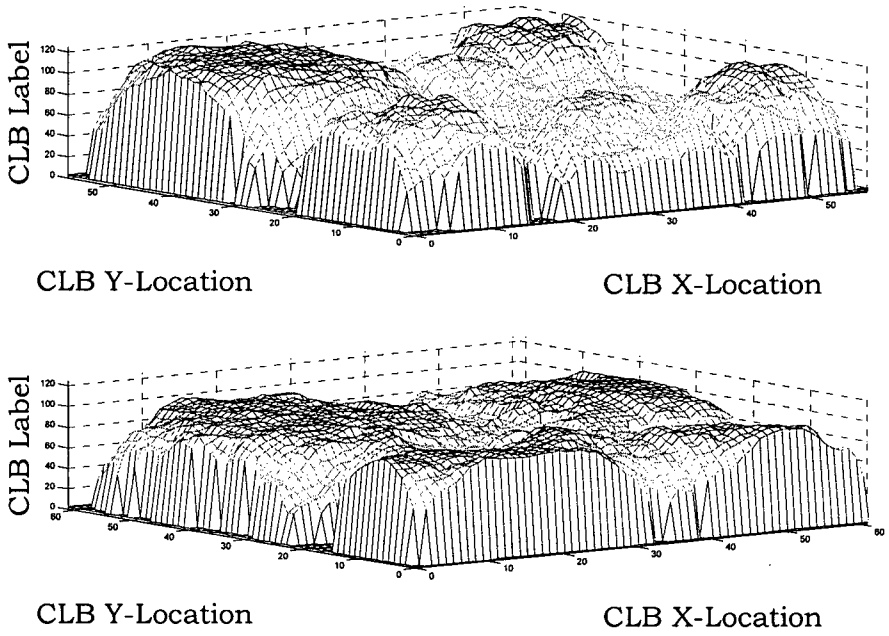

Figure 6-2: Congestion Map Before and After Un/DoPack

From Figure 6-2, it is observed that both the peak and variation of CLB labels were both decreased.

The depopulation region center is determined by finding the largest label in the congestion map. In the case of a tie, the CLB that is closest to the center of the map is chosen as the depopulation center. Two different methods were considered to determine how large the depopulation region is and how much to depopulate the region by.

1. **Coarse Grained:** A circle is drawn around the depopulation center with a radius of 1/4 the logical dimension of the array. For example, in a 19x19 CLB grid, the region radius would be floor(1/4*19)=4. All CLBs within the drawn circle are inserted into the depopulation region. The new BLE-limit size is

determined such that the increase in the total number of CLBs will fill an entire new row/column in the *entire array.*

2. **Fine Grained:** A circle is drawn around the depopulation center with a possible radius of 1/4, 1/5, 1/6 or 1/8 the logical dimension of the array. The new cluster size is determined so that the increase in the total number of CLBs will fill an entire row/column in *just the depopulation region.*

The coarse grained approach increases the array size by one in the x and y direction in every iteration. In comparison, the fine grained approach grows the array size much more slowly as there is no guarantee that enough new CLBs will be created to fill an entire new row/column in the array after each iteration.

6.1.3 Step 3: DoPack - Incremental Re-Cluster

The third step (DoPack) performs incremental re-clustering of the depopulated region with the smaller BLE-limit size constraint. It re-clusters the BLEs from the depopulation region identified by UnPack and leaves all other CLBs outside of this region untouched. UnPack provides DoPack with the new BLE-limit size limit to use, which guarantees the production of more CLBs. This is crucial: by using more CLBs, the congested region can span more routing channels to obtain more total routing tracks. This tool can use any existing clustering method (*e.g.* T-VPack [6], T-RPack [8], iRAC [46]) as the underlying packing engine since the only changing constraint is the BLE-limit size.

6.1.4 Step 4: Placement and Routing

Finally, the fourth step produces a new placed and routed solution. The purpose of the place and route steps is to accurately identify regions of routing congestion. Ideally, this could be done with a fast congestion estimator that can precisely locate the regions of peak routing demand before placement is done. Unfortunately, no such tool exists for FPGAs. Meanwhile, Un/DoPack uses actual place and route directly; this is slow, but accurate. If the channel width constraint is not met, Steps 2-4 are iterated until the given channel width constraint is met. Due to iteration in the flow, it is important to speed up both the placement and routing steps as much as possible. These options are discussed below.

6.1.4.1 Faster Placement

To speed up placement, VPR was modified to perform incremental placement. *The work is currently being performed by David Leong in the UBC SoC research group.* The incremental placer was compared to VPR's builtin "–fast" mode. The incremental placer attempts to preserve the placement locations of CLBs outside of the depopulation region. It provides placement stability by preserving the previous placement solution as much as possible. This should not only decrease run time, but also provide consistent and predictable changes as the CAD flow iterates to reduce channel width. The incremental placer works in stages. The first stage is an "expansion" phase which squeezes the numerous "depopulated" CLBs into the "too small" space left behind. This produces illegal solutions when CLBs are pushed

outside of the array bounds. The second stage is a "compaction" phase used to legalize the solution. The third stage is an optional low-temperature anneal to clean up the solution. The output of the incremental algorithm is a solution that is computed using a fraction of the time required for a full placement. Development of the incremental placer continues to be an on-going process.

6.1.4.2 Faster Routing

To speed up routing, we attempted to obtain congestion results from the first iterations of the VPR Pathfinder routing algorithm. At this stage, there is significant illegal wire sharing. Because wire sharing is not heavily penalized, most nets will take the shortest paths from source to sink. These failed routing solutions do not represent the final congestion regions so this data was not very useful. Since this data could not be used, the VPR router was allowed to run to completion which is the primary reason why this approach is slow. No attempts to develop an incremental routing algorithm have been made yet.

6.2 Experimental Results

This section presents the performance results of running Un/DoPack on the two benchmark circuit suites (Meta and GNL). The first set of benchmarks is the Meta benchmark suite presented in Section 4.1. The second set of benchmarks is the Stdev benchmark suite presented in Section 4.2. The baseline flow of Un/DoPack uses the following options:

- **UnPack:** Coarse grained congestion calculated (Section 6.1.2).

- **DoPack:** A replica of the iRAC algorithm as the underlying clustering algorithm (Section 6.1.3).
- **Fast Placement:** Incremental placer described in section 6.1.4.1.
- **Fast Routing:** None. Use fully routed solution.

Because of large run times and limited computing resources, a maximum run time of 48 hours is imposed. If the time limit is exceeded, Un/DoPack concludes that no solution exists. All computations were performed on a dedicated Intel Xeon, 3GHz processor with 1.5GB of RAM. Before Un/DoPack was run on a benchmark circuit, VPR was first used to determine the MRCW of a circuit (with no depopulation). This was done by invoking the binary search option of the VPR router. This is the **maximum channel width constraint** for each circuit. Then, Un/DoPack was run with various channel width constraints up to 45% below the maximum channel width constraint. Since these channel width constraints are below the maximum, some amount of depopulation must occur to meet the given channel width constraint. The next sub-section presents experimental results after running Un/DoPack on the Stdev benchmark suite. The subsequent sub-section compares the performance of Un/DoPack to DHPack using the Meta benchmark suite.

6.2.1 Stdev Benchmark Circuit Results

This section presents the experimental results for the baseline flow of Un/DoPack on the Stdev benchmark suite. These results will demonstrate that Un/DoPack is effective at reducing MRCW. The results will also show that the

effectiveness (amount of area inflation) of Un/DoPack at reducing MRCW is dependent on the amount of interconnect variation in the benchmark circuit. The choices for the baseline flow is then justified by a series of experiments designed to test placement stability and congestion region selection.

6.2.1.1 Baseline Flow Results

Table 6-1 shows the maximum achievable channel width reductions (normalized to the case where no depopulation was required) Un/DoPack was able to generate within CPU time limit and array size constraints.

Circuit	Max. Channel Width Change	Critical Path Change	Area Change	# of iterations
Stdev000	-34%	+12%	+91%	30
Stdev002	-42%	+22%	+79%	30
Stdev004	-35%	+21%	+68%	25
Stdev006	-32%	+17%	+86%	28
Stdev008/clone	-33%	+17%	+60%	26
Stdev010	-48%	+22%	+39%	23
Stdev012	-35%	+13%	+25%	16
Arithmetic Mean	**-38%**	**+18%**	**+64%**	**25**

Table 6-1: Maximum % Change in Channel Width, Critical Path and Area

On average, channel width decreases of 38% was possible for the set of Stdev benchmark circuits with an 18% penalty in critical path delay and a 64% increase in total area required. A complete table of results is given in Appendix C.

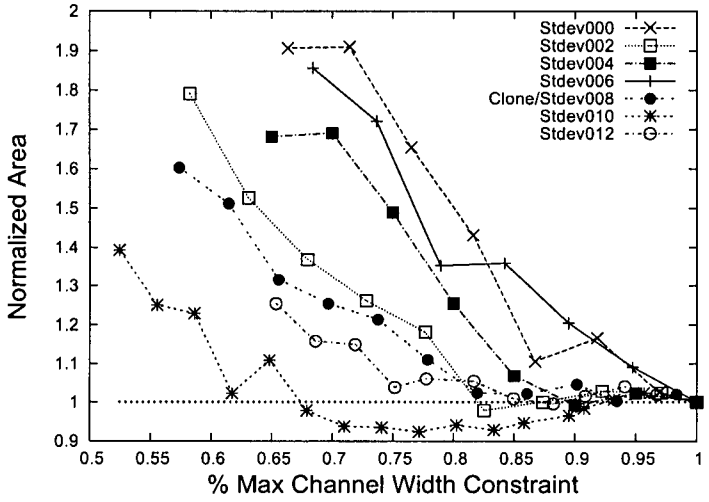

Figure 6-3: Normalized Area vs. % Max Channel Width Constraint

Figure 6-3 shows the normalized area increase for each Stdev circuit as the channel width constraint is decreased (e.g. more depopulation is applied). In general, area tends to decrease due to lower channel widths, but increase due to more CLBs. Therefore a total area increase indicates that the area added by more CLBs exceeds the savings in area due to a shrink in channel width.

Figure 6-3 also suggests that the amount of interconnect variation in a circuit affects the effectiveness of Un/DoPack. For circuits with high interconnect variation (Stdev010, Stdev012), significant channel width savings is possible with virtually no area inflation. Circuits with low interconnect variation (Stdev004, Stdev002, Stdev000) show quick area increases with modest channel width reduction. For example, circuit Stdev010 shows a 40% decrease in channel width with only 10% increase in area. This occurs because there is a very local high congestion region and only a small amount of depopulation is needed to reduce the congestion in this region.

In contrast, circuit Stdev000 shows large area increases for small decreases in channel width (e.g. 25% channel width decrease, 90% area increase). This suggests that a large amount of depopulation is needed to reduce the channel width of a circuit that has a uniformly distributed congestion map. Note that the ordering of the curves in Figure 6-3 does not exactly match the standard deviation of the circuit. For example, Stdev006 has higher area increases than Stdev002. This is likely due to the variation in the Stdev benchmark circuits. GNL is random in nature and although a Rent parameter can be specified for each IP block, there is no guarantee that Rent parameter was achieved. Interconnect variation in a circuit is also not solely defined by the standard deviation of the Rent parameters of the IP blocks. Many other factors including placement and routing constraints may affect the amount of interconnect variation in a circuit. The important result from Figure 6-3 is that the general trend indicates that circuits with low interconnect variation require large area increases to reduce channel width and circuits with high interconnect variation require smaller area increases for channel width reduction.

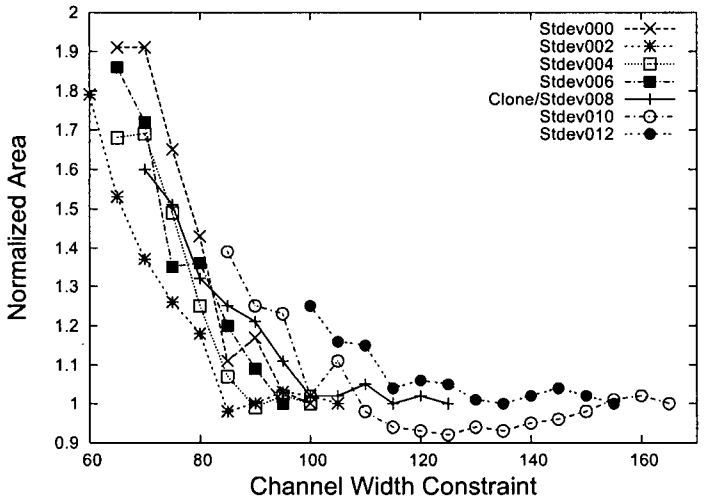

Figure 6-4: Normalized Area vs. Absolute Channel Width Constraint

Figure 6-4 shows the same data as Figure 6-3 but the x-axis shows the absolute channel widths instead of the normalized channel width. Note that circuits with high interconnect variation require significantly higher absolute channel widths to route (without constraints). This suggests that it is crucial for FPGA architects to know the *amount of interconnect variation* within their benchmark circuits. If the variation is too high, it is possible that the routing networks will be designed with excess capacity, resulting in undue cost to the consumer. Fortunately, these very circuits are the most amenable to channel width reduction using the Un/DoPack flow.

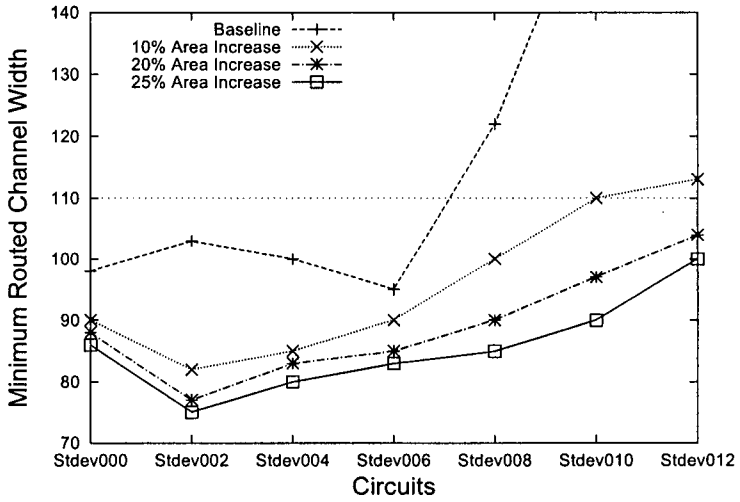

Figure 6-5: MRCW vs. Stdev Circuit

Another view of the data from Figure 6-3 and Figure 6-4 is given in Figure 6-5. Figure 6-5 shows the channel widths that were attainable with the baseline flow (with no constraints) and for constraints that produce net area increases of 10%, 20% and 25%. FPGA architects typically choose channel widths for their devices to fit the most number of circuits possible. If an architect were to choose a channel width for a specific device with only the unconstrained baseline results, a channel width greater than 140 tracks would be chosen so that all the circuits would have a routable solution. However, a more realistic choice for the channel width of the device may be 110 tracks. This would result in a 21% (= 1–110/140) decrease in channel width which translates directly into a significant area savings. The few circuits that could not be mapped to such a device could then be depopulated to meet the given channel width constraint. Although this is a simplistic method for an architect to choose a channel width for a device, it does highlight that being able to quantify the amount of interconnect variation in the circuit is important in FPGA design.

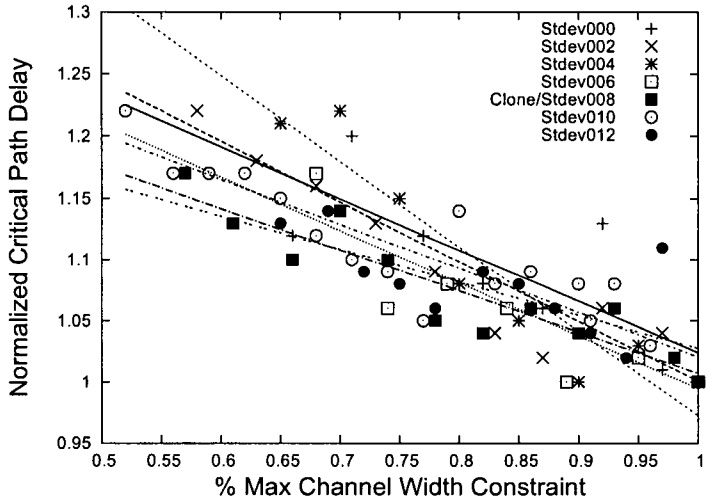

Figure 6-6: Critical Path Delay vs. Channel Width Constraint

The other trade-off for channel width reduction is an increase in critical path delay. Figure 6-6 shows the normalized critical path vs. channel width constraint expressed as a percentage of the maximum channel width. Since the data points have significant noise, trendlines (least squares fit) are shown for each circuit. There is on average a 10% penalty in critical path for a 20% decrease in channel width, and a 23% increase in critical path for a 45% decrease in channel width. Most likely, designs that have very strict timing constraints would likely use high performance FPGAs which have high-capacity routing networks (e.g. Altera Stratix II). Channel width reduction techniques such as DHPack and Un/DoPack are intended for low-cost FPGA families (e.g. Altera Cyclone II) where designs are not as timing critical.

Figure 6-7 shows the run times of Un/DoPack, which are dominated by the slow routing step. Run times quickly increase for channel width reductions of greater than 20%, but are relatively small for channel width constraints below 20%. On

average, a 20% channel width reduction is possible with approximately 200% increase in run time. Figure 6-7 shows that run times increase more quickly for circuits with low interconnect variation (eg. Stdev000) than circuits with high variation (Stdev010). Also shown are run-times of Stdev010 with fine-grained depopulation. These experiments are explained in the next sub-section.

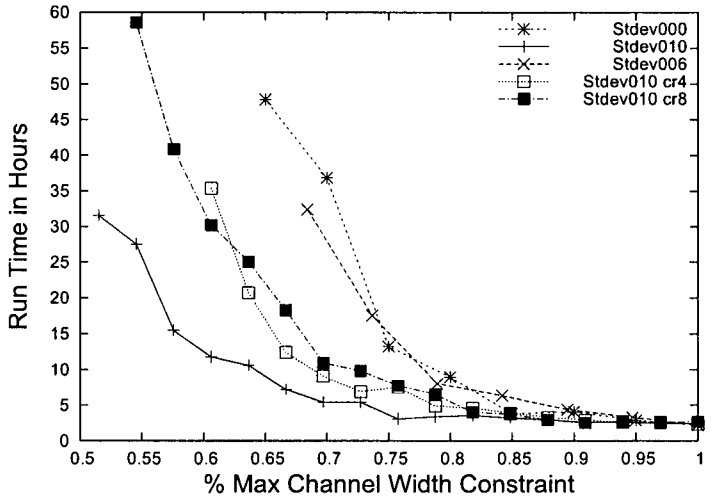

Figure 6-7: Run Times vs. Channel Width Constraint

6.2.1.2 Placement Stability and Congestion Region Experiments

Three different experiments were performed to determine the suitability of some of the choices made in the baseline flow. These experiments are designed to test the effects of placement stability and congestion region estimation. The three experiments are described below.

1. **PlaceScratch:** After Un/DoPack finishes, the final clustered solution for each channel width constraint is re-placed and re-routed using the default VPR placer. Since the default VPR placer starts with a completely random initial

placement, all of the placement stability maintained during the iterative process will be lost. This test determines if the critical path is degraded by many calls to the incremental placer.

2. **VPR –fast:** Un/DoPack was re-run with the VPR –fast option as the fast placement engine instead of the incremental placer. The VPR –fast option begins with a random initial placement which means no placement stability is maintained in the iterative loop of Un/DoPack. This test seeks to find the effect of placement stability on run time and whether the congested regions "move around" unpredictably.

3. **Fine Grained Congestion Estimation:** Un/DoPack was re-run using the fine grained technique described in section 6.1.2. This test seeks to determine if performing depopulation more slowly can lead to superior area results.

Experiment	Description	Geometric Mean of Normalized Area	Geometric Mean of Normalized CP	Geometric Mean of Total Run Time	# channel width constr. w/ no sol'n (74 total)
0	Baseline	1.00	1.00	1.00	0
1	PlaceScratch	1.00	0.98	N/A	18
2	-fast	1.01	0.99	0.93	3
3	cr4	0.98	1.00	1.55	10
	cr5	0.96	0.99	1.39	8
	cr6	0.95	1.01	1.48	6
	cr8	0.94	0.99	1.65	11

Table 6-2: Results for PlaceScratch, -fast and Fine Grained

Table 6-2 shows the simulation results for each of the 3 experiments. In Experiment 3, cr4 to cr8 indicates fine grained congestion estimation was used with a congestion radius of 1/4 to 1/8 the logical dimension of the array and the maximum run time was increased from 48 hours to 72 hours.

The data in Table 6-2 is calculated as follows. There are 74 total channel width constraints across the 7 different Stdev benchmark circuits. These 74 channel width constraints were chosen because the baseline flow was able to find solutions for these channel width constraints under the time limit constraints (e.g. 48 hours). These constraints ranged from 0% to 45% below the maximum channel width constraint for each circuit. However, these channel width constraints could not always be met for a given experiment. For example, experiment 3 (cr6) was not able to generate solutions for 6 of the 74 channel width constraints because the time limit of 72 hours was exceeded. For experiment 3 (cr6) ,the averages reported in Table 6-2 represents the geometric mean of normalized results for all channel width constraints excluding the 6 constraints that could not be met. This same method is used to calculate all of the data in Table 6-2. In most cases, the channel width constraints that are not met are the "*aggressive*" channel width constraints (e.g. more than 30% below maximum channel width constraint). Hence, the results may be somewhat conservative considering the "*aggressive*" channel width constraints are often excluded from the reported results.

Table 6-2 shows that performing a placement and route from scratch on the final clustered netlist shows a modest 2% decrease in critical path delay. This suggests

that the incremental placement engine is generating high quality placements. The last column in Table 6-2 is a measure of the stability of the Baseline flow. When placing the final clustered netlist from scratch, 18 out of the 72 circuits could no longer meet the given channel width constraint. This is because the incremental placement provides stability in the channel width reduction process. By re-placing the entire circuit from scratch, the circuit is disturbed enough such that the channel width constraint an no longer be met. This suggests that the incremental placer is important to preserve placement stability during the iterative channel width reduction process.

Table 6-2 shows that using the "–fast" option in VPR generates similar results in terms of area and critical path delay with a 7% decrease in run time. This suggests that the "–fast" option is generating good placements and that placement stability present in the incremental placer is not as important as originally hypothesized.

Table 6-2 also shows that using fine grained congestion estimation improves area at the expense of increased run time. Area reductions of 2%, 4%, 5% and 6% are possible for cr4, cr5, cr6 and cr8 congestion radii respectively with a run time penalty of 55%, 39%, 48% and 65%. Once again, the final column shows that although superior area results are achievable using fine grained congestion estimation, the ability of Un/DoPack to converge to a solution is also affected. For example, experiment cr8 could not converge to a solution in 11 out of the 74 channel width constraints in the 72 hour time limit. Figure 6-7 also highlights that fine grained congestion estimation significantly increases run time since a larger number of

iterations needs to be performed to converge to a solution. Future work is needed to tune the Un/DoPack congestion region choices and run times.

6.2.2 Comparison of Un/DoPack and DHPack

This section will compare the performance of Un/DoPack to DHPack. It was not possible to run DHPack using the Stdev circuits because the precise design hierarchy of the Stedev benchmarks is unknown. Therefore, the comparison in this section is limited to the Meta benchmark circuits. The Meta benchmark suite was created stitching IP blocks together post-clustering. Since Un/DoPack has no partitioning requirement, the Meta benchmarks were stitched together pre-clustering. Functionally, the new Meta benchmarks are identical to the old set. The difference is the new set gives Un/DoPack the ability to merge BLEs from different IP blocks together. Un/DoPack was run for all 3 Meta benchmark circuits: Clique, Pipeline, and Independent.

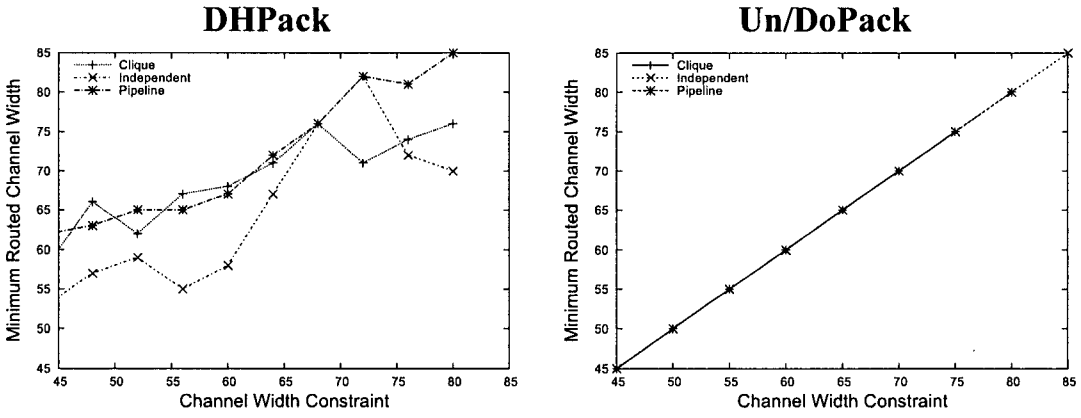

Figure 6-8: MRCW for DHPack vs. Un/DoPack

Figure 6-8 shows the MRCW versus the channel width constraint for DHPack and Un/DoPack. Figure 6-8 highlights that although DHPack can target a channel width constraint, many solutions are generated that exceed the constraint. In contrast, Un/DoPack consistently meets the given channel width constraint.

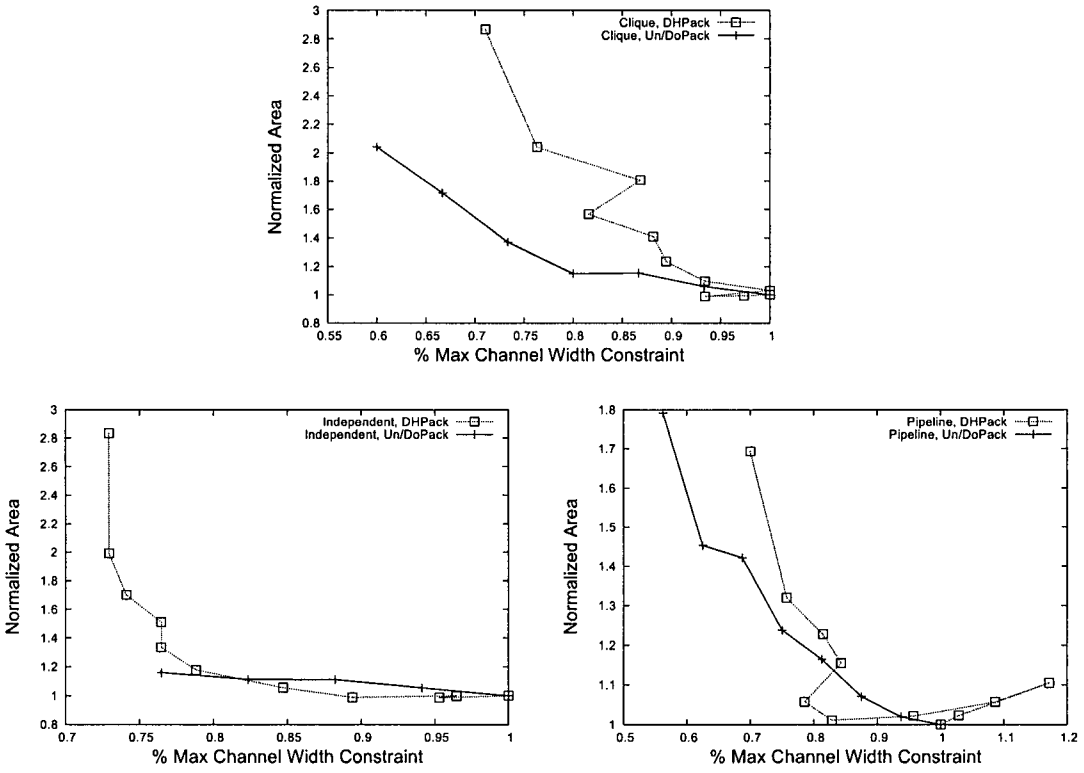

Figure 6-9: Comparison of Area between DHPack and Un/DoPack

Figure 6-9 show an area comparison between DHPack and Un/DoPack for the Meta benchmark circuits. In these graphs, the x-axis represents the routed channel width expressed as a percentage of the maximum channel width constraint. The DHPack curves appear a bit erratic in Figure 6-9 because even if a channel width constraint is specified, there is no guarantee that DHPack will meet that given channel

width constraint; in many cases, the routed channel width was greater than the channel width constraint. Figure 6-9 suggests that DHPack produces similar (but less predictable) area results for small channel width reductions whereas Un/DoPack generates superior area results for more aggressive channel width reductions. Future work includes tuning Un/DoPack to more accurately identify congestion regions for modest channel width reductions.

6.3 Experimental Conclusions

This chapter has presented an iterative CAD tool (Un/DoPack) for channel width reduction. Compared to DHPack, Un/DoPack does not have the requirement that the design hierarchy be known a priori. Un/DoPack identifies a congestion region and iteratively depopulates this region until a given channel width constraint is met. On average, a channel width reduction of 38% was achievable with an 18% penalty in critical path delay and a 64% increase in area for the Stdev benchmark circuits.

Un/DoPack highlighted the importance of the interconnect variation metric for channel width reduction. Since large circuits are not available to the academic community, it is difficult for researchers to know how much interconnect variation exists in real world circuits. The Stdev benchmark circuits address this by providing a set of circuits that have a wide range of interconnect variation from virtually no interconnect variation (Stdev000) to an extreme amount of variation (Stdev012). The experimental results demonstrate that a large amount of depopulation is necessary to reduce channel width in circuits with low interconnect variation. Conversely, a small

amount of depopulation is needed to reduce channel width in circuits with high interconnect variation.

The amount of interconnect variation also has consequences to FPGA architecture design. It was found that circuits with high interconnect variation tend to have large absolute channel widths. Being able to quantify the amount of interconnect variation is important for FPGA architects who may unnecessarily over design their routing networks.

Finally, a comparison between the single pass technique DHPack and the multi pass technique Un/DoPack shows that Un/DoPack is much more effective at meeting hard channel width constraints. This is an important property in practice as mapping to industrial FPGAs requires hard channel width constraints. Results show that Un/DoPack produces superior area results for larger channel width constraints but may benefit from more tuning when only small channel width reductions are needed.

6.4 Technique Limitations and Future Work

This section will discuss some of the limitations of Un/DoPack and some possible directions for future work.

6.4.1 Fast Placement Improvements

The biggest disadvantage of Un/DoPack is the long run times associated with "aggressive" channel width constraints. The large portion of the run time is consumed by the placement and routing portions of the iterative flow. Two possible methods to

improve the fast placement process are to improve the incremental placement tool or to use a congestion-driven placer.

6.4.1.1 Incremental Placement

It was originally hypothesized that the incremental placer would decrease run times by decreasing placement time and decreasing the number of iterations required to converge to a solution. Unfortunately, the incremental placer did not perform much better than the VPR –fast option in terms of total run time. The advantage of the incremental placer is placement stability. Figure 6-2 shows that the congestion peaks remain in the same relative locations using the incremental placer. Generating similar congestion profiles using the –fast option in VPR does not show this type of stability. Future work involves using the placement stability feature of the incremental placer to more quickly converge to a solution and improving incremental placement speed.

6.4.1.2 Congestion Driven Placement

It should also be noted that VPR placement is wirelength-driven, not congestion-driven. Congestion-driven placement tools such as [9] or [44] attempt to reduce interconnect variation by finding placements which evenly distribute interconnect demand across the circuit. These tools perform this by using congestion metrics in the placement cost function rather than enforcing depopulation. The use of congestion aware placement tools along with a depopulation strategy should be explored in the future. It may reduce the number of iterations needed by Un/DoPack to converge to a solution.

6.4.2 Benchmark Interconnect Variation Verification

It is very difficult to predict how much interconnect variation is present in current industrial designs. The Stdev benchmark circuits attempt to address this by presenting a set of benchmark circuits that have a range of interconnect variation. However, the Stdev circuits are still synthetic in nature and may not be entirely representative of real circuits. Since the performance of Un/DoPack is dependent on the amount of interconnect variation and the size or granularity of different congestion regions, it is important to measure the amount of interconnect variation in real circuits. Real world designs should be analyzed to determine the extent of interconnect variation within each design. Studies have been published on Rent parameter variation across different designs [41] but not within a single design. Future work includes running Un/DoPack on commercial SoC designs and measuring the amount of interconnect variation within these circuits.

Chapter 7

7 CONCLUSION AND FUTURE WORK

Despite the over-design in the capacity of routing networks in FPGAs, there may still be circuits that remain un-routable in a specific FPGA family. The traditional solution to this problem was to switch to the next higher performance FPGA family (e.g. Altera Cyclone [2] to Altera Stratix[3]) resulting in significantly higher monetary costs. This thesis has shown that rather than migrating to the next FPGA family, a circuit can be made routable by using a larger FPGA device in the same family. This is performed by trading-off logic utilization for channel width. This thesis has presented two CAD tools (DHPack and Un/DoPack) capable of performing this trade-off so that a circuit can be mapped to a channel width constrained FPGA. In particular, the methods help fit hard-to-route circuits into FPGAs with limited interconnect. It does this at the expense of using more CLBs. Since larger devices with more CLBs are usually available, this is a practical trade-off. Experimental results have demonstrated that it is sufficient to selectively depopulate parts of a large circuit that would otherwise have routing congestion. Only the routing intensive portions of the circuit are depopulated to keep area inflation to a minimum. A summary of the maximum channel width reductions for DHPack and Un/DoPack are given in Table 7-1 and Table 7-2.

Circuit	Clustering Tool	Channel Width and Area Changes			
		CW	Area	CW	Area
Clique	T-VPack	-19%	+6%	-50%	+129%
	iRAC Rep.	-7%	-1%	-29%	+187%
Independent	T-VPack	-25%	+2%	-55%	+184%
	iRAC Rep.	-17%	+1%	-30%	+69%
Pipeline	T-VPack	-24%	+3%	-42%	+184%
	iRAC Rep.	-15%	+6%	-27%	+183%
Arithmetic Mean	T-VPack	**-23%**	**+4%**	**-49%**	**+166%**
	iRAC Rep.	**-13%**	**+3%**	**-29%**	**+146%**

Table 7-1: Summary of Channel Width Decreases for DHPack

Circuit	Channel Width Change	Critical Path Change	Area Change
Stdev000	-34%	+12%	+91%
Stdev002	-42%	+22%	+79%
Stdev004	-35%	+21%	+68%
Stdev006	-32%	+17%	+86%
Stdev008/clone	-33%	+17%	+60%
Stdev010	-48%	+22%	+39%
Stdev012	-35%	+13%	+25%
Arithmetic Mean	**-38%**	**+18%**	**+64%**

Table 7-2: Summary of Channel Width Decreases for Un/DoPack

DHPack results show that non-uniform depopulation is important for keeping area inflation to a minimum during channel width reduction. DHPack is able to achieve, on average, MRCW decreases of 23% / 13% for T-VPack / iRAC with 4% / 3% increases in area. On average, larger MRCW decreases of 39% / 29% are possible with 166% / 146% increase in area.

On average, Un/DoPack can target channel width constraints 38% below the max channel width constraint with an 18% penalty in critical path delay and 64% increase in area. Un/DoPack results have shown that channel width reduction is "easy"

for circuits with high interconnect variation whereas channel width reduction is "hard" for circuits with low interconnect variation.

Most importantly, both DHPack and Un/DoPack have demonstrated that un-routable circuits can be made routable by buying an FPGA with more logic!

7.1 Future Work

This section will summarize the future work already presented in Section 5.4 for DHPack and Section 6.4 for Un/DoPack. Some discussion into other possible methods for congestion estimation and improved FPGA modeling is then provided.

7.1.1 DHPack Future Work

Section 5.4 discusses the limitations and possible future work for DHPack in more detail. This future work is summarized below.

- **I/O Padframe Congestion:** The MRCW did not track the channel width constraint in some cases because of congestion in the channel adjacent to the I/O padframe. Doubling the channel width of the I/O channels relative to the rest of the chip may help alleviate this problem. Another possible solution is to adjust the placement algorithm to be aware of localized congestion, especially at the I/O periphery.
- **IP Block Granularity Too Coarse:** DHPack relies on a pre-defined IP block partitioning to identify congestion regions. This strict partitioning requirement does not allow the exploration of other possible partitioning

methods which may more accurately identify congestion regions. A congestion method that does not rely on the design hierarchy was presented in Chapter 6 (Un/DoPack).

- **Hard Channel Width Constraints:** DHPack may generate MRCW results greater than the channel width constraint. For practical purposes, channel width reduction tools must be able to target hard channel width constraints. This issue was addressed in Chapter 6 (Un/DoPack).
- **Congestion Profile Run Time Long**: Creating the congestion profile for each IP block can be a time consuming process. Each possible BLE-limit size for each IP block must be individually placed and routed. Faster methods of computing congestion were suggested in Chapter 6 (Un/DoPack) and is discussed at the end of this chapter.

7.1.2 Un/DoPack Future Work

Section 6.4 discusses the limitations and possible future work for Un/DoPack in more detail. This future work is summarized below.

- **Fast Placement Improvements:** The largest contributor to the long run times of Un/DoPack is the iterative placement and routing step. An incremental placement tool (developed by David Leong at UBC) attempts to preserve placement stability to improve run time. However, experimental results showed that VPR "-fast" option runs just as quickly. Work continues in the UBC SoC group to use the placement stability of the incremental placer to

improve run times. Use of a congestion driven placement tool ([9], [44]) may also improve MRCW and area results by more evenly distributing interconnect demand and reducing interconnect variation. It may also improve Un/DoPack run-time if fewer iterations are required.

- **Benchmark Interconnect Variation:** Since it is unknown how much interconnect variation exists in real SoC designs, the Stdev benchmarks were created to represent a set of circuits with a wide range of interconnect variation. However, the Stdv benchmark circuits are still synthetic in nature and may not accurately represent commercial SoC designs. It is important to quantify how much interconnect variation exists in real world circuit and to evaluate Un/DoPack on these circuits.

7.1.3 System Level Interconnect Prediction

Because of the orthogonal nature of the different steps of the FPGA CAD flow, designers must often go back and forth between synthesis, placement and routing to satisfy user specified criteria (e.g. timing constraints). To help alleviate the large effort typically used to meet these constraints, the use of a priori interconnect estimation methods have been widely studied to help predict which regions of a circuit will be difficult to route. This information is then used to help drive the synthesis, placement and routing process. These same methods could also be used to help determine whether a given hard channel width constraint can be met without having to do a complete place and route. Some of these techniques include probabilistic methods [35], post-placement methods [25], and a priori (pre-placement) methods [5]. These

techniques should be further investigated to determine whether the run time of the iterative portion of the Un/DoPack flow can be improved.

7.1.4 Improved FPGA Modeling

The FPGA model used in this thesis consisted of programmable logic elements and programmable routing elements.

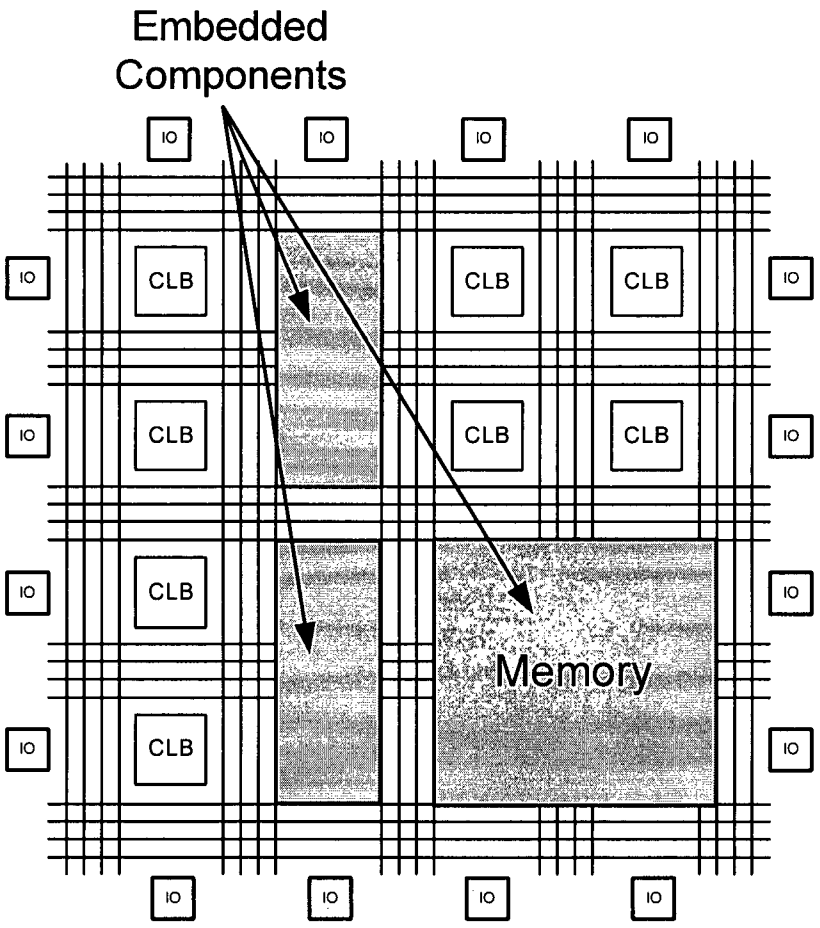

Figure 7-1: FPGA Architecture with Macro Blocks

However, most commercial FPGAs contain an increasingly larger number of hard macro blocks. These macro blocks can include embedded memories, multipliers, or high speed I/Os. Figure 7-1 shows an example of FPGA architecture with macro blocks. The channel width reduction techniques described in this thesis assumes that logic can be spread out uniformly across the FPGA. However, in an FPGA architecture with macro blocks, the logic blocks do not have as much freedom to migrate to other locations. I/O connections to the macro blocks can also be a source of routing congestion. Since depopulation of macro blocks is not possible, other methods to reduce congestion around the periphery of these blocks needs to be investigated.

8 REFERENCES

[1] E. Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density", ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp 3-12, 2000.

[2] Altera Corp. Cyclone FPGA Family Overview. http://www.altera.com/products/devices/cyclone/overview/cyc-overview.html

[3] Altera Corp. Stratix FPGA Family Overview. http://www.altera.com/products/devices/stratix/overview/stx-overview.html

[4] J. Anderson and F. Najm, "Power-Aware Technology Mapping for LUT-based FPGAs", IEEE International Conference on Field Programmable Technology, pp. 211-218, 2002.

[5] S. Balachandran, "A-Priori Interconnect Estimation for Field Programmable Gate Arrays", PhD Dissertation, University of Texas at Dallas, 2005.

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

[7] E. Bozorgzadah et al, "RPack: Routability-Driven Packing for Cluster-Based FPGAs", IEEE Asia South Pacific Design Automation Conference, 2001.

[8] E. Bozorgzadeh et al, "Routability-Driven Packing : Metrics and Algorithms for Cluster-based FPGAs", IEEE Journal of Circuits, Systems, and Computers, Vol. 13, No. 1, pp. 77-100, 2004.

[9] U. Brenner, A. Rohe, "An Effective Congestion Driven Placement Framework", ACM International Symposium on Physical Design, pp.6-11, 2002.

[10] Y. Chang, S. Thakur, K. Zhu, and D. Wong, "A New Global Routing Algorithm for FPGAs", IEEE/ACM International Conference on Computer Aided Design, pp. 356-361, 1994.

[11] D. Chen, et al, "Low-Power Technology Mapping for FPGA Architectures with dual Supply Voltages", ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 109-117, 2004.

[12] J. Cong and Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs", IEEE Transactions on Computer-Aided Design, pp. 1-12, 1994.

[13] J. Cong and Y. Ding, "On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping", IEEE Transactions on VLSI Systems, Vol. 2, No. 2, pp. 137-148, 1994.

[14] J. Cong, and Y. Hwang, "Simultaneous Depth and Area Minimization in LUT-Based FPGA Mapping", ACM/SIGDA International Symposium on Field-Programmable Gate Array, pp. 68-74, 1995.

[15] J. Cong, and Y. Hwang, "Structural Gate Decomposition for Depth-Optimal Technology in LUT-based FPGA Designs", ACM Transactions on Design Automation of Electronic Systems, Vol. 5, No. 3, 2000.

[16] M. Dehkordi and S. D. Brown, "The Effect of Cluster Packing and Node Duplication Control in Delay Driven Clustering", IEEE International Conference on Field Programmable Technology, pp. 227-233, 2002

[17] A. DeHon, "Balancing Interconnect and Computation in a Reconfigurable Computing Array", ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 69-78, 1999.

[18] A. Dunlop and B. Kernighan, "A Procedure for Placement of Standard-Cell VLSI Circuits", IEEE Transactions on Computer-Aided Design, pp. 92-98, 1985.

[19] Future Electronics E-Store. http://www.futureestore.com/eStore/Default.aspx

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

[21] P. Hallschmid, "Detailed Routing Architectures for Embedded Programmable IP Cores", M.A.Sc. Thesis, University of British Columbia, 2003.

[22] D.J.H Huang and A.B. Kahng, "When Clusters Meet Partitions: New Density-Based Methods for Circuit Decomposition", IEEE European Design and Test Conference, pp. 60-64, 1995.

[23] M. Hutton, J. Rose, and D. Corneil, "Automatic Generation of Synthetic Sequential Benchmark Circuits", IEEE Transactions on Computer-Aided Design, Vol. 21, No. 8, 2002.

[24] D. Huang and A. Kahng, "Partitioning-Based Standard-Cell Global Placement with an Exact Objective", ACM Symposium on Physical Design, pp. 18-25, 1997.

[25] P. Kannan, S. Shankar and D. Bhatia, "fGREP – Fast Generic Routing Demand Estimation for Placed FPGA Circuits", International Conference on Field Programmable Logic and Applications, pp. 37-47 , 2001.

[26] A. Kennings and I. L. Markov, "Analytical Minimization of Half-perimeter Wire Length", IEEE Asia and South Pacific Design Automation Conference, pp. 179-184, 2000.

[27] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing", Science, Vol. 220, No. 4698, pp. 671-680, 1983.

[28] P. Kundarewich and J. Rose, "Synthetic Circuit Generation Using Clustering and Iteration", IEEE Transactions on Computer-Aided Design, Vol. 23, No. 6, 2004.

[29] J. Lamoureux, "On the Interaction between Power-Aware Computer-Aided Design Algorithms for Field-Programmable Gate Arrays", M.A.Sc. Thesis, University of British Columbia, 2003.

[30] B.Landman and R. Russo, "On a Pin Versus Block Relationship for Partitions of Logic Graphs", IEEE Transactions on Computers, Vol. C-20, pp. 1469-1479, 1971.

[31] G. Lemieux, S. Brown, "A Detailed Router for Allocating Wire Segments in FPGAs", ACM Physical Design Workshop, pp. 215-226, 1993.

[32] G. Lemieux, S. Brown, D. Vranesic, "On Two-Step Routing for FPGAs", ACM Symposium on Physical Design, pp. 60-66, 1997.

[33] G. Lemieux, E. Lee, M. Tom and A. Yu, "Directional and Single-Driver Wiring in FPGA Interconnect", IEEE International Conference on Field-Programmable Technology, pp. 41-48, 2004.

[34] P. Leventis, M. Chan, et al, "Cyclone: A Low-Cost, High-Performance FPGA", IEEE Custom Integrated Circuits Conference, pp. 49-52, 2003.

[35] J. Lou, S. Krishnamoorthy, and H. Sheng, "Estimating Routing Congestion Using Probabilistic Analysis", ACM International Symposium on Physical Design, pp 112-117, 2001.

[36] LGSynth93 Benchmark Suite, "Microelectronics Centre of North Carolina", Tech. Report, 1993.

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

[38] A. Marquardt, V. Betz, and J. Rose, "Timing-Driven Placement for FPGAs", ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 203-213, 2000.

[39] L. McMurchie, and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs", ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 111-117, 1995.

[40] R. Murgai, R. Brayton and A. Sangiavanni-Vincentelli, "On Clustering for Minimum Delay/Area", IEEE International Conference on Computer Aided Design, pp.6-9, 1991.

[41] J. Pistorius and M. Hutton, "Placement Rent Exponent Calculation Methods, Temporal Behaviour and FPGA Architecture Evaluation", ACM International Workshop on System-Level Interconnect Prediction, pp. 31-38, 2003.

[42] B. Riess, K. Doll, and F. Johannes, "Partitioning Very Large Circuits Using Analytical Placement Techniques", ACM/SIGDA Design Automation Conference, pp. 646-651, 1994.

[43] J. Rose, W. Snelgrove and Z. Vranesic, "ALTOR: An Automatic Standard Cell Layout Program", Proceedings of Canadian Conference on VLSI, pp. 169-173, 1985.

[44] A. Sharma, C. Ebeling and S. Hauck, "Architecture-Adaptive Routability-Driven Placement for FPGAs", International Conference on Field Programmable Logic, pp. 427-432, 2005.

[45] G. Sigl, K. Doll and F. Johannes, "Analytical Placement: A Linear or Quadratic Objective Function?", ACM/SIGDA Design Automation Conference, pp. 427-432, 1991.

[46] A. Singh and M. Marek-Sadowska, "Efficient Circuit Clustering for Area and Power reduction in FPGAs", International Symposium on Field Programmable Gate Arrays, pp. 59-66, 2002.

[47] D. Tavana, W. Lee, S. Young and B. Fawcett, "Logic Block and Routing Considerations for a New SRAM-Based FPGA Architecture", IEEE Custom Integrated Circuits Conference, pp. 24.6.1 – 24.6.4, 1995.

[48] R. Tessier and H. Giza, "Balancing Logic Utilization and Area Efficiency in FPGAs", International Workshop on Field Programmable Logic and Applications, pp. 535-544, 2000.

[49] M. Tom and G. Lemieux, "Logic Block Clustering of Large Designs for Channel-Width Constrained FPGAs", ACM/SIGDA Design Automation Conference, pp. 726-731, 2005.

[50] M. Tom, D. Leong and G. Lemieux, "Un/DoPack: Re-Clustering of Large System-on-Chip Designs with Interconnect Variation for Low-Cost FPGAs", Submitted to ACM/SIGDA Design Automation Conference 2006.

[51] P. Verplaetse, D. Stroobandt, and J. van Campenhout, "Synthetic Benchmark Circuits for Timing-Driven Physical Design Applications", Proceedings of International Conference on VLSI, pp 31-37, 2002.

[52] Y.-L. Wu and M. Marek-Sadowska, "An Efficient Router for 2-D Field-Programmable Gate Arrays", Proceedings of European Design Automation Conference, pp. 412-416, 1994.

[53] Xilinx Corp. Spartan-3 FPGA Family Overview. http://www.xilinx.com/products/silicon_solutions/fpgas/spartan_series/spartan3_fpgas/index.htm

[54] H. Yang and D. Wong, "Circuit Clustering for Delay Minimization Under Area and Pin Constraints", IEEE Transactions on Computer Aided Design, Vol. 16, No. 9, pp. 976-986, 1997.

Appendix A – Stdev Benchmark Circuit Parameters

	Rent Parameters (Stdev)							
IP Block	000	002	004	006	008	010	012	Num BLEs
bigkey.blif	0.62	0.58	0.55	0.51	0.48	0.44	0.40	915
dsip.blif	0.62	0.59	0.56	0.52	0.49	0.46	0.42	912
s38584.1.blif	0.62	0.60	0.57	0.55	0.52	0.50	0.48	5411
tseng.blif	0.62	0.60	0.58	0.56	0.54	0.52	0.50	1182
elliptic.blif	0.62	0.61	0.59	0.58	0.56	0.55	0.53	3255
s38417.blif	0.62	0.61	0.59	0.58	0.56	0.55	0.53	4555
diffeq.blif	0.62	0.61	0.59	0.58	0.57	0.56	0.54	1245
des.blif	0.62	0.61	0.60	0.59	0.58	0.57	0.57	554
s298.blif	0.62	0.61	0.61	0.60	0.60	0.59	0.58	1309
frisc.blif	0.62	0.62	0.62	0.61	0.61	0.61	0.61	3814
alu4.blif	0.62	0.63	0.63	0.64	0.65	0.65	0.66	1173
clma.blif	0.62	0.63	0.63	0.64	0.65	0.66	0.66	6273
misex3.blif	0.62	0.63	0.65	0.66	0.67	0.69	0.70	1158
spla.blif	0.62	0.64	0.65	0.66	0.68	0.69	0.71	3005
pdc.blif	0.62	0.64	0.65	0.67	0.69	0.70	0.72	3629
seq.blif	0.62	0.64	0.66	0.68	0.70	0.71	0.73	1325
ex5p.blif	0.62	0.64	0.66	0.68	0.70	0.72	0.75	740
apex2.blif	0.62	0.64	0.67	0.69	0.72	0.74	0.77	1478
ex1010.blif	0.62	0.65	0.67	0.70	0.73	0.75	0.78	3093
apex4.blif	0.62	0.65	0.67	0.70	0.73	0.75	0.78	969

Appendix B – DHPack Simulation Results

DHPack Non-Uniform Depopulation (T-VPack)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	average channel width
Clique					
95	93	1.41E+08	2.35E-08	2550	83.14
90	82	1.35E+08	2.55E-08	2576	79.34
85	85	1.39E+08	3.09E-08	2605	80.21
80	79	1.38E+08	3.55E-08	2692	76.08
75	77	1.41E+08	2.39E-08	2770	73.95
70	75	1.49E+08	2.30E-08	2939	71.44
65	73	1.65E+08	3.46E-08	3268	67.33
60	69	1.90E+08	2.34E-08	3882	62.49
55	57	2.15E+08	2.34E-08	4651	54.14
50	52	2.44E+08	2.37E-08	5404	50.55
Pipeline					
95	102	1.53E+08	2.34E-08	2550	88.26
90	90	1.48E+08	2.36E-08	2576	85.21
85	93	1.48E+08	2.92E-08	2605	88.21
80	82	1.46E+08	2.32E-08	2692	78.52
75	82	1.53E+08	2.23E-08	2770	78.57
70	76	1.57E+08	2.27E-08	2939	74.61
65	73	1.66E+08	2.30E-08	3268	70.14
60	64	1.91E+08	2.36E-08	3882	62.37
55	55	2.29E+08	2.38E-08	4651	54.35
50	56	2.62E+08	2.33E-08	5404	53.19
Independent					
95	106	1.47E+08	2.40E-08	2550	94.24
90	99	1.40E+08	2.39E-08	2576	92.21
85	92	1.46E+08	2.63E-08	2605	86.27
80	87	1.41E+08	2.31E-08	2692	83.20
75	89	1.45E+08	2.23E-08	2770	83.21
70	81	1.50E+08	2.62E-08	2939	76.68
65	71	1.65E+08	2.26E-08	3268	67.78
60	66	1.87E+08	2.77E-08	3882	63.95
55	67	2.13E+08	2.48E-08	4651	59.41
50	65	2.48E+08	2.35E-08	5404	56.55

DHPack Non-Uniform Depopulation (iRAC Replica)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	average channel width
Clique					
80	76	1.30E+08	2.67E-08	2544	71.41
76	74	1.29E+08	2.73E-08	2553	70.84
72	71	1.28E+08	2.86E-08	2588	68.63
68	76	1.34E+08	2.87E-08	2604	71.41
64	71	1.42E+08	2.67E-08	2853	66.33
60	68	1.60E+08	2.53E-08	3273	63.03
56	67	1.83E+08	2.57E-08	3765	59.22
52	62	2.03E+08	2.59E-08	4273	54.96
48	66	2.35E+08	2.50E-08	4897	55.81
44	58	2.65E+08	2.62E-08	5758	48.56
40	54	3.72E+08	2.67E-08	8219	43.59
Pipeline					
80	70	2.13E+08	2.77E-08	2544	60.02
76	72	2.18E+08	2.69E-08	2553	61.05
72	82	2.36E+08	2.87E-08	2588	63.91
68	76	2.26E+08	2.67E-08	2604	63.10
64	67	2.18E+08	2.69E-08	2853	57.98
60	58	2.16E+08	2.58E-08	3273	53.19
56	55	2.26E+08	2.58E-08	3765	51.38
52	59	2.46E+08	2.52E-08	4273	52.78
48	57	2.62E+08	2.52E-08	4897	50.06
44	53	2.81E+08	2.56E-08	5758	47.12
40	49	3.61E+08	2.63E-08	8219	42.16
Independent					
80	85	1.38E+08	2.57E-08	2544	78.29
76	81	1.36E+08	2.70E-08	2553	76.97
72	82	1.38E+08	2.80E-08	2588	76.59
68	76	1.37E+08	2.68E-08	2604	72.82
64	72	1.46E+08	2.68E-08	2853	69.78
60	67	1.63E+08	2.52E-08	3273	64.29
56	65	1.84E+08	2.59E-08	3765	60.30
52	65	2.09E+08	2.50E-08	4273	58.62
48	63	2.35E+08	2.56E-08	4897	54.96
44	62	2.75E+08	2.54E-08	5758	50.34
40	62	3.91E+08	2.57E-08	8219	43.75

DHPack Uniform Depopulation (T-VPack)				
channel width constraint	**routed channel width**	**area (#min size transistors)**	**critical path (s)**	**# of CLBs**
Clique				
95	106	1.48E+08	2.91E-08	2539
90	105	1.56E+08	3.97E-08	2709
85	97	1.57E+08	4.94E-08	2901
80	91	1.76E+08	4.68E-08	3385
75	86	1.83E+08	4.32E-08	3692
70	83	1.96E+08	5.09E-08	4062
65	74	2.29E+08	3.66E-08	5084
60	62	2.79E+08	5.64E-08	6776
55	51	3.82E+08	6.92E-08	10172
50	44	4.87E+08	1.19E-07	13500
45	40	7.07E+08	6.96E-08	20250
Pipeline				
95	101	1.46E+08	2.55E-08	2539
90	99	1.52E+08	2.74E-08	2708
85	95	1.57E+08	4.53E-08	2901
80	87	1.73E+08	2.79E-08	3384
75	85	1.83E+08	4.62E-08	3692
70	78	1.93E+08	5.23E-08	4064
65	74	2.29E+08	3.70E-08	5082
60	63	2.80E+08	5.16E-08	6775
55	51	3.83E+08	4.78E-08	10172
50	43	4.85E+08	6.53E-08	13500
45	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Independent				
95	104	1.51E+08	3.31E-08	2539
90	99	1.56E+08	3.46E-08	2708
85	100	1.63E+08	2.84E-08	2901
80	87	1.76E+08	2.98E-08	3386
75	87	1.88E+08	4.01E-08	3692
70	79	1.97E+08	5.39E-08	4063
65	67	2.25E+08	5.56E-08	5084
60	65	2.86E+08	2.75E-08	6778
55	64	4.06E+08	2.49E-08	10172
50	60	5.21E+08	2.33E-08	13500
45	57	7.54E+08	3.03E-08	20250

DHPack Uniform Depopulation (iRAC Replica)				
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs
Clique				
80	79	1.31E+08	7.07E-08	2533
76	77	1.35E+08	4.49E-08	2702
72	69	1.59E+08	5.09E-08	3378
68	73	1.73E+08	6.87E-08	3685
64	61	1.95E+08	5.49E-08	4507
60	62	2.17E+08	4.33E-08	5071
56	51	2.66E+08	4.74E-08	6763
52	45	3.05E+08	1.09E-07	8125
48	45	3.05E+08	1.09E-07	8125
44	43	3.70E+08	7.94E-08	10158
40	39	4.77E+08	1.17E-07	13500
Pipeline				
80	81	1.33E+08	2.77E-08	2533
76	80	1.37E+08	3.28E-08	2702
72	70	1.60E+08	3.33E-08	3378
68	69	1.70E+08	3.39E-08	3685
64	62	1.97E+08	4.07E-08	4507
60	60	2.15E+08	4.22E-08	5071
56	51	2.66E+08	4.69E-08	6764
52	47	3.08E+08	3.25E-08	8124
48	47	3.08E+08	3.25E-08	8124
44	45	3.74E+08	6.77E-08	10159
40	42	4.84E+08	6.98E-08	13500
Independent				
80	84	1.38E+08	3.90E-08	2533
76	82	1.42E+08	3.40E-08	2702
72	72	1.65E+08	3.45E-08	3378
68	68	1.73E+08	4.38E-08	3685
64	65	2.02E+08	4.17E-08	4507
60	64	2.22E+08	2.72E-08	5071
56	63	2.83E+08	2.66E-08	6764
52	63	3.32E+08	2.66E-08	8125
48	63	3.32E+08	2.66E-08	8125
44	62	4.03E+08	2.74E-08	10160
40	61	5.23E+08	2.64E-08	13500

Appendix C – Un/DoPack Simulation Results

Un/DoPack (Baseline)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev000					
100	98	1.98E+08	7.14E-08	3148	7695
95	95	2.03E+08	7.23E-08	3349	10199
90	90	2.31E+08	8.11E-08	3892	14630
85	85	2.19E+08	7.61E-08	3820	13637
80	80	2.84E+08	7.70E-08	5067	32156
75	75	3.28E+08	8.01E-08	6061	47542
70	70	3.79E+08	8.56E-08	7214	132652
65	65	3.78E+08	8.01E-08	7394	172100
Stdev002					
105	103	2.04E+08	6.79E-08	3157	6995
100	100	2.08E+08	7.08E-08	3360	9937
95	95	2.09E+08	7.17E-08	3428	11617
90	90	2.03E+08	6.92E-08	3424	11298
85	85	1.99E+08	7.07E-08	3473	11085
80	80	2.40E+08	7.37E-08	4310	17509
75	75	2.57E+08	7.68E-08	4728	25406
70	70	2.78E+08	7.87E-08	5272	36858
65	65	3.11E+08	7.97E-08	6057	83070
60	60	3.64E+08	8.29E-08	7339	117923
Stdev004					
100	100	1.98E+08	6.87E-08	3148	8184
95	95	2.03E+08	7.10E-08	3328	9163
90	90	1.97E+08	6.87E-08	3310	8290
85	85	2.12E+08	7.21E-08	3681	13711
80	80	2.49E+08	7.42E-08	4479	21905
75	75	2.95E+08	7.86E-08	5455	36421
70	70	3.35E+08	8.39E-08	6374	60015
65	65	3.33E+08	8.34E-08	6487	97740
Stdev006					
95	95	1.94E+08	7.22E-08	3139	8043
90	90	2.11E+08	7.37E-08	3571	11731
85	85	2.33E+08	7.23E-08	4067	15749
80	80	2.63E+08	7.68E-08	4727	22742
75	75	2.62E+08	7.78E-08	4786	28865
70	70	3.33E+08	7.62E-08	6306	63077
65	65	3.59E+08	8.42E-08	7011	116596

Un/DoPack (Baseline)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev008 / clone					
125	122	2.23E+08	7.15E-08	3151	8278
120	120	2.27E+08	7.27E-08	3333	9147
115	114	2.23E+08	7.60E-08	3357	8767
110	110	2.33E+08	7.41E-08	3593	11125
105	105	2.28E+08	7.57E-08	3575	11482
100	100	2.28E+08	7.40E-08	3644	13311
95	95	2.47E+08	7.54E-08	4086	19057
90	90	2.70E+08	7.88E-08	4578	25888
85	85	2.79E+08	8.16E-08	4891	37240
80	80	2.93E+08	7.89E-08	5248	55563
75	75	3.37E+08	8.06E-08	6225	104358
70	70	3.57E+08	8.37E-08	6738	145958
Stdev010					
165	162	2.61E+08	7.08E-08	3152	9013
160	155	2.67E+08	7.29E-08	3335	9740
155	151	2.63E+08	7.64E-08	3341	11463
150	147	2.57E+08	7.42E-08	3326	11033
145	145	2.52E+08	7.66E-08	3312	11153
140	139	2.47E+08	7.70E-08	3333	11255
135	135	2.42E+08	7.61E-08	3325	11972
130	130	2.45E+08	8.07E-08	3451	13388
125	125	2.41E+08	7.41E-08	3459	14454
120	120	2.44E+08	7.73E-08	3595	17004
115	115	2.44E+08	7.78E-08	3638	21693
110	110	2.55E+08	7.96E-08	3915	27715
105	105	2.89E+08	8.18E-08	4516	37099
100	100	2.67E+08	8.30E-08	4281	47924
95	95	3.21E+08	8.29E-08	5307	68951
90	90	3.26E+08	8.29E-08	5500	89670
85	85	3.63E+08	8.63E-08	6360	127204
Stdev012					
155	153	2.52E+08	7.50E-08	3163	11323
150	148	2.56E+08	8.30E-08	3320	12837
145	144	2.62E+08	7.62E-08	3480	15609
140	139	2.56E+08	7.77E-08	3461	14954
135	135	2.51E+08	7.94E-08	3451	15965
130	130	2.54E+08	8.10E-08	3579	20082
125	125	2.65E+08	8.16E-08	3813	24351
120	119	2.67E+08	7.94E-08	3917	29451
115	115	2.61E+08	8.12E-08	3909	31113
110	110	2.89E+08	8.20E-08	4473	47925
105	105	2.92E+08	8.52E-08	4597	63338
100	100	3.16E+08	8.44E-08	5060	117460

Un/DoPack (PlaceScratch)					
channel width constraint	**routed channel width**	**area (#min size transistors)**	**critical path (s)**	**# of CLBs**	**run time (s)**
Stdev000					
100	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
95	95	2.03E+08	7.15E-08	3349	N/A
90	90	2.31E+08	7.14E-08	3892	N/A
85	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
80	80	2.84E+08	7.59E-08	5067	N/A
75	75	3.28E+08	7.60E-08	6061	N/A
70	70	3.79E+08	8.10E-08	7214	N/A
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
Stdev002					
105	103	2.04E+08	6.77E-08	3157	N/A
100	99	2.08E+08	6.89E-08	3360	N/A
95	94	2.09E+08	7.19E-08	3428	N/A
90	90	2.03E+08	6.98E-08	3424	N/A
85	85	1.99E+08	7.15E-08	3473	N/A
80	80	2.40E+08	7.45E-08	4310	N/A
75	75	2.57E+08	7.15E-08	4728	N/A
70	70	2.78E+08	7.37E-08	5272	N/A
65	65	3.11E+08	7.63E-08	6057	N/A
60	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
Stdev004					
100	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
95	94	2.03E+08	7.13E-08	3328	N/A
90	90	1.97E+08	7.26E-08	3310	N/A
85	85	2.12E+08	7.14E-08	3681	N/A
80	80	2.49E+08	7.28E-08	4479	N/A
75	75	2.95E+08	7.47E-08	5455	N/A
70	70	3.35E+08	7.51E-08	6374	N/A
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
Stdev006					
95	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
90	90	2.11E+08	7.00E-08	3571	N/A
85	85	2.33E+08	7.82E-08	4067	N/A
80	80	2.63E+08	7.30E-08	4727	N/A
75	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
70	70	3.33E+08	7.75E-08	6306	N/A
65	65	3.59E+08	8.25E-08	7011	N/A

Un/DoPack (PlaceScratch)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev008 / clone					
125	123	2.23E+08	7.08E-08	3151	N/A
120	118	2.27E+08	6.93E-08	3333	N/A
115	115	2.23E+08	7.01E-08	3357	N/A
110	109	2.33E+08	6.98E-08	3593	N/A
105	104	2.28E+08	7.05E-08	3575	N/A
100	100	2.28E+08	7.58E-08	3644	N/A
95	95	2.47E+08	7.26E-08	4086	N/A
90	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
85	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
80	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
75	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
70	70	3.57E+08	7.80E-08	6738	N/A
Stdev010					
165	163	2.61E+08	7.20E-08	3152	N/A
160	154	2.67E+08	7.27E-08	3335	N/A
155	147	2.63E+08	7.50E-08	3341	N/A
150	145	2.57E+08	7.87E-08	3326	N/A
145	143	2.52E+08	7.45E-08	3312	N/A
140	139	2.47E+08	7.29E-08	3333	N/A
135	135	2.42E+08	7.51E-08	3325	N/A
130	129	2.45E+08	7.44E-08	3451	N/A
125	124	2.41E+08	7.43E-08	3459	N/A
120	119	2.44E+08	7.50E-08	3595	N/A
115	114	2.44E+08	7.49E-08	3638	N/A
110	110	2.55E+08	7.47E-08	3915	N/A
105	105	2.89E+08	7.69E-08	4516	N/A
100	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
95	95	3.21E+08	8.33E-08	5307	N/A
90	90	3.26E+08	7.90E-08	5500	N/A
85	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
Stdev012					
155	155	2.52E+08	7.60E-08	3163	N/A
150	148	2.56E+08	8.37E-08	3320	N/A
145	142	2.62E+08	8.44E-08	3480	N/A
140	137	2.56E+08	8.71E-08	3461	N/A
135	135	2.51E+08	7.41E-08	3451	N/A
130	129	2.54E+08	7.54E-08	3579	N/A
125	124	2.65E+08	7.71E-08	3813	N/A
120	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
115	115	2.61E+08	7.68E-08	3909	N/A
110	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
105	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A
100	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	N/A

Un/DoPack (--fast)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev000					
100	98	1.98E+08	7.14E-08	3148	7898
95	95	2.09E+08	7.27E-08	3423	9678
90	90	2.32E+08	7.29E-08	3918	15691
85	85	2.25E+08	7.25E-08	3901	16422
80	80	2.95E+08	7.60E-08	5329	30228
75	75	3.42E+08	7.66E-08	6265	67524
70	70	4.01E+08	7.99E-08	7572	115332
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev002					
105	103	2.04E+08	6.79E-08	3157	7520
100	98	2.08E+08	7.19E-08	3360	8613
95	95	2.04E+08	6.86E-08	3359	7910
90	90	2.18E+08	7.16E-08	3670	11531
85	85	2.12E+08	7.56E-08	3683	11604
80	80	2.34E+08	7.32E-08	4204	16649
75	75	2.47E+08	7.39E-08	4523	26645
70	70	3.26E+08	7.88E-08	6197	54553
65	65	3.52E+08	7.75E-08	6881	92238
60	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev004					
100	100	1.98E+08	6.87E-08	3148	7856
95	95	2.03E+08	7.28E-08	3328	7805
90	90	2.05E+08	7.06E-08	3467	8616
85	85	2.19E+08	7.33E-08	3796	11303
80	80	2.34E+08	7.22E-08	4189	17412
75	75	2.47E+08	7.56E-08	4520	19596
70	70	3.50E+08	7.64E-08	6624	46381
65	65	3.60E+08	7.65E-08	7038	61765
Stdev006					
95	95	1.94E+08	7.22E-08	3139	7113
90	90	1.97E+08	7.12E-08	3318	7442
85	85	2.26E+08	7.34E-08	3932	14061
80	80	2.42E+08	8.03E-08	4350	15446
75	75	2.49E+08	7.67E-08	4576	18592
70	70	3.51E+08	8.26E-08	6665	65325
65	65	3.76E+08	8.07E-08	7337	71711

Un/DoPack (--fast)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev008 / clone					
125	122	2.23E+08	7.15E-08	3151	8781
120	118	2.27E+08	7.13E-08	3333	9990
115	113	2.23E+08	7.13E-08	3357	8753
110	110	2.16E+08	7.37E-08	3310	8108
105	105	2.13E+08	7.43E-08	3334	8271
100	100	2.30E+08	7.40E-08	3713	12261
95	95	2.45E+08	7.23E-08	4012	16803
90	90	2.84E+08	7.58E-08	4780	25825
85	85	3.11E+08	7.60E-08	5431	49176
80	80	2.92E+08	7.79E-08	5228	52517
75	75	2.88E+08	7.75E-08	5316	68675
70	70	3.61E+08	8.16E-08	6871	119092
Stdev010					
165	162	2.61E+08	7.08E-08	3152	8653
160	155	2.67E+08	7.18E-08	3335	8605
155	152	2.63E+08	7.21E-08	3341	9248
150	145	2.57E+08	7.28E-08	3326	9362
145	145	2.52E+08	7.93E-08	3312	10544
140	140	2.47E+08	7.40E-08	3333	11555
135	135	2.42E+08	7.67E-08	3325	12727
130	128	2.45E+08	7.42E-08	3436	12252
125	125	2.33E+08	7.39E-08	3360	11105
120	120	2.60E+08	7.49E-08	3838	19516
115	115	2.46E+08	7.56E-08	3693	19576
110	110	2.46E+08	8.25E-08	3761	25958
105	105	2.92E+08	8.34E-08	4606	38219
100	100	2.76E+08	8.25E-08	4463	42320
95	95	2.86E+08	7.97E-08	4732	55761
90	90	3.38E+08	8.12E-08	5742	99029
85	85	3.55E+08	8.55E-08	6234	113725
Stdev012					
155	153	2.52E+08	7.50E-08	3163	9588
150	148	2.56E+08	7.74E-08	3320	10721
145	142	2.70E+08	7.54E-08	3586	16121
140	138	2.56E+08	8.52E-08	3463	13294
135	134	2.51E+08	7.70E-08	3449	14334
130	130	2.54E+08	7.79E-08	3589	17002
125	125	2.56E+08	8.82E-08	3660	20452
120	120	3.01E+08	7.72E-08	4428	38847
115	115	2.62E+08	8.07E-08	3935	38892
110	110	3.05E+08	8.44E-08	4671	69455
105	105	3.17E+08	8.48E-08	5010	77107
100	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*

Un/DoPack (Fine Grain cr4)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev000					
100	98	1.98E+08	7.14E-08	3148	8471
95	95	2.02E+08	7.21E-08	3300	12317
90	90	2.38E+08	7.43E-08	3998	26777
85	85	2.38E+08	7.31E-08	4101	36121
80	80	2.68E+08	7.96E-08	4783	62678
75	75	2.84E+08	8.02E-08	5186	105735
70	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev002					
105	103	2.04E+08	6.79E-08	3157	7978
100	100	2.00E+08	7.25E-08	3198	8270
95	95	2.02E+08	7.02E-08	3313	11501
90	90	2.02E+08	7.17E-08	3389	13099
85	85	2.12E+08	7.17E-08	3678	18578
80	80	2.32E+08	7.25E-08	4140	35017
75	75	2.55E+08	7.68E-08	4682	50968
70	70	2.92E+08	8.34E-08	5489	155298
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
60	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev004					
100	100	1.98E+08	6.87E-08	3148	8234
95	95	2.02E+08	7.28E-08	3294	9309
90	90	1.95E+08	6.99E-08	3261	9283
85	85	1.91E+08	7.21E-08	3318	12600
80	80	2.39E+08	7.38E-08	4249	34324
75	75	2.55E+08	7.37E-08	4652	51740
70	70	3.02E+08	7.73E-08	5710	101079
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev006					
95	95	1.94E+08	7.22E-08	3139	7134
90	90	2.16E+08	7.18E-08	3625	14185
85	85	2.12E+08	7.24E-08	3685	21663
80	80	2.54E+08	9.49E-08	4529	35402
75	75	2.77E+08	7.67E-08	5069	70393
70	70	3.11E+08	7.50E-08	5918	143285
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*

Un/DoPack (Fine Grain cr4)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev008 / clone					
125	122	2.23E+08	7.15E-08	3151	7509
120	120	2.25E+08	7.26E-08	3257	10320
115	115	2.21E+08	7.87E-08	3278	10641
110	110	2.25E+08	7.73E-08	3479	16077
105	105	2.20E+08	7.55E-08	3470	15926
100	100	2.36E+08	7.44E-08	3792	25533
95	95	2.32E+08	7.46E-08	3820	29578
90	90	2.32E+08	7.83E-08	3902	33066
85	85	2.79E+08	7.92E-08	4866	74683
80	80	3.01E+08	7.85E-08	5381	131647
75	75	3.35E+08	7.93E-08	6157	228512
70	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev010					
165	162	2.61E+08	7.08E-08	3152	8183
160	159	2.57E+08	7.31E-08	3189	9400
155	154	2.53E+08	7.73E-08	3190	9244
150	149	2.48E+08	7.78E-08	3223	10372
145	143	2.51E+08	7.40E-08	3289	11724
140	140	2.46E+08	7.76E-08	3295	14032
135	135	2.43E+08	8.10E-08	3361	16368
130	130	2.44E+08	7.99E-08	3404	17554
125	125	2.48E+08	7.57E-08	3549	26880
120	120	2.42E+08	8.16E-08	3550	24782
115	115	2.45E+08	7.66E-08	3641	32658
110	110	2.40E+08	8.42E-08	3705	44565
105	105	2.67E+08	7.97E-08	4225	74664
100	100	2.85E+08	8.19E-08	4622	127301
95	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
90	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
85	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev012					
155	153	2.52E+08	7.50E-08	3163	11326
150	148	2.47E+08	8.08E-08	3195	11960
145	144	2.50E+08	7.59E-08	3269	14902
140	140	2.53E+08	7.92E-08	3369	19045
135	135	2.42E+08	7.72E-08	3336	18308
130	130	2.54E+08	7.69E-08	3579	29567
125	125	2.63E+08	8.58E-08	3728	43890
120	120	2.60E+08	8.03E-08	3842	61118
115	115	2.78E+08	8.17E-08	4171	89943
110	110	3.07E+08	8.63E-08	4743	158373
105	105	3.01E+08	8.25E-08	4754	188420
100	100	2.99E+08	8.01E-08	4786	198160

Un/DoPack (Fine Grain cr5)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev000					
100	98	1.98E+08	7.14E-08	3148	7308
95	95	2.02E+08	7.30E-08	3291	9844
90	90	2.16E+08	7.48E-08	3604	20066
85	85	2.26E+08	7.35E-08	3943	27829
80	80	2.79E+08	7.54E-08	5035	59611
75	75	3.11E+08	8.05E-08	5726	153841
70	70	3.32E+08	7.94E-08	6250	210952
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev002					
105	103	2.04E+08	6.79E-08	3157	8614
100	100	2.00E+08	6.95E-08	3211	7978
95	95	2.01E+08	7.28E-08	3257	8735
90	90	2.03E+08	7.21E-08	3405	14688
85	85	2.10E+08	7.05E-08	3615	18542
80	80	2.27E+08	7.59E-08	4078	36797
75	75	2.40E+08	8.05E-08	4366	51391
70	70	2.70E+08	7.33E-08	5094	112252
65	65	2.99E+08	7.86E-08	5782	183348
60	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev004					
100	100	1.98E+08	6.87E-08	3148	6747
95	95	2.01E+08	6.97E-08	3282	10872
90	90	1.97E+08	6.97E-08	3332	9901
85	85	1.97E+08	7.27E-08	3415	11875
80	80	2.24E+08	7.33E-08	3984	30918
75	75	2.63E+08	7.57E-08	4829	81675
70	70	2.86E+08	7.62E-08	5423	88617
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev006					
95	95	1.94E+08	7.22E-08	3139	6560
90	90	2.11E+08	7.62E-08	3568	17801
85	85	2.10E+08	7.16E-08	3615	17388
80	80	2.46E+08	7.20E-08	4374	36987
75	75	2.63E+08	7.41E-08	4820	72897
70	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
65	65	3.02E+08	7.86E-08	5868	191051

Un/DoPack (Fine Grain cr5)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev008 / clone					
125	122	2.23E+08	7.15E-08	3151	7151
120	120	2.19E+08	7.40E-08	3203	8066
115	115	2.16E+08	7.29E-08	3238	11009
110	110	2.15E+08	7.33E-08	3261	12225
105	105	2.11E+08	7.64E-08	3276	10467
100	100	2.28E+08	7.55E-08	3664	26911
95	95	2.18E+08	7.56E-08	3588	26431
90	90	2.48E+08	7.78E-08	4193	49063
85	85	2.63E+08	7.89E-08	4578	82963
80	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
75	75	3.09E+08	7.75E-08	5670	222197
70	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev010					
165	162	2.61E+08	7.08E-08	3152	7852
160	159	2.57E+08	7.41E-08	3204	8832
155	152	2.53E+08	7.36E-08	3206	8858
150	150	2.48E+08	7.72E-08	3203	9002
145	144	2.44E+08	7.74E-08	3243	12513
140	140	2.46E+08	7.52E-08	3297	12264
135	135	2.42E+08	7.63E-08	3317	13467
130	130	2.36E+08	7.99E-08	3299	16727
125	125	2.41E+08	7.48E-08	3459	23626
120	120	2.41E+08	7.49E-08	3490	28776
115	115	2.44E+08	7.79E-08	3614	31842
110	110	2.47E+08	8.19E-08	3770	46464
105	105	2.59E+08	7.77E-08	4077	62333
100	100	2.58E+08	8.13E-08	4141	83755
95	95	2.92E+08	8.27E-08	4790	165006
90	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
85	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev012					
155	153	2.52E+08	7.50E-08	3163	9160
150	148	2.48E+08	8.23E-08	3209	12823
145	144	2.50E+08	7.70E-08	3265	12630
140	140	2.39E+08	8.27E-08	3244	13346
135	135	2.41E+08	7.98E-08	3283	17650
130	130	2.44E+08	7.77E-08	3392	21991
125	125	2.50E+08	8.17E-08	3598	40364
120	120	2.49E+08	8.07E-08	3634	60756
115	115	2.63E+08	8.20E-08	3966	76521
110	110	2.78E+08	8.35E-08	4243	132110
105	105	3.09E+08	8.73E-08	4890	200707
100	100	3.16E+08	8.56E-08	5068	210189

Un/DoPack (Fine Grain cr6)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev000					
100	98	1.98E+08	7.14E-08	3148	8465
95	95	1.95E+08	7.38E-08	3192	9961
90	90	2.04E+08	7.52E-08	3450	15171
85	85	2.24E+08	7.37E-08	3855	33693
80	80	2.47E+08	7.21E-08	4427	57179
75	75	2.80E+08	7.59E-08	5153	93545
70	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev002					
105	103	2.04E+08	6.79E-08	3157	7052
100	100	2.00E+08	6.97E-08	3201	9611
95	95	2.01E+08	6.94E-08	3276	11680
90	90	2.10E+08	7.69E-08	3516	17716
85	85	2.11E+08	7.28E-08	3647	23903
80	80	2.12E+08	7.22E-08	3778	29967
75	75	2.64E+08	8.53E-08	4851	90594
70	70	2.69E+08	7.52E-08	5063	95327
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
60	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev004					
100	100	1.98E+08	6.87E-08	3148	6650
95	95	1.95E+08	6.92E-08	3192	7396
90	90	1.97E+08	7.09E-08	3335	10676
85	85	2.17E+08	7.46E-08	3760	22456
80	80	2.26E+08	7.45E-08	4048	32613
75	75	2.25E+08	7.41E-08	4102	44358
70	70	2.84E+08	7.75E-08	5336	114758
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev006					
95	95	1.94E+08	7.22E-08	3139	6836
90	90	2.02E+08	7.07E-08	3367	11580
85	85	2.18E+08	8.32E-08	3790	27344
80	80	2.45E+08	7.88E-08	4362	55837
75	75	2.41E+08	7.40E-08	4422	51874
70	70	3.08E+08	8.10E-08	5804	176043
65	65	3.25E+08	8.11E-08	6306	236774

Un/DoPack (Fine Grain cr6)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev008 / clone					
125	122	2.23E+08	7.15E-08	3151	7515
120	119	2.19E+08	7.68E-08	3195	8111
115	115	2.15E+08	7.11E-08	3208	8971
110	110	2.25E+08	7.35E-08	3462	17234
105	105	2.20E+08	7.30E-08	3458	15662
100	100	2.19E+08	7.72E-08	3488	19644
95	95	2.24E+08	7.38E-08	3698	27897
90	90	2.45E+08	8.46E-08	4112	50205
85	85	2.53E+08	7.81E-08	4380	80109
80	80	2.70E+08	7.68E-08	4836	139538
75	75	3.00E+08	8.58E-08	5494	212419
70	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev010					
165	162	2.61E+08	7.08E-08	3152	8185
160	157	2.57E+08	7.93E-08	3196	9297
155	153	2.53E+08	7.46E-08	3196	9347
150	148	2.47E+08	7.58E-08	3195	10095
145	144	2.44E+08	7.65E-08	3225	10892
140	140	2.45E+08	7.79E-08	3263	11996
135	134	2.40E+08	7.62E-08	3266	14322
130	130	2.36E+08	7.81E-08	3292	16738
125	125	2.41E+08	7.75E-08	3462	22348
120	120	2.41E+08	7.96E-08	3503	28773
115	115	2.36E+08	8.13E-08	3498	28329
110	110	2.41E+08	8.19E-08	3717	48848
105	105	2.41E+08	7.76E-08	3773	53242
100	100	2.60E+08	8.38E-08	4200	104875
95	95	2.62E+08	8.20E-08	4321	113149
90	90	2.70E+08	8.54E-08	4570	163037
85	85	2.79E+08	8.05E-08	4890	212494
Stdev012					
155	153	2.52E+08	7.50E-08	3163	9603
150	149	2.48E+08	8.20E-08	3202	11319
145	144	2.51E+08	7.74E-08	3277	15534
140	138	2.47E+08	7.96E-08	3325	16067
135	135	2.43E+08	8.18E-08	3349	21361
130	130	2.45E+08	7.84E-08	3452	25725
125	125	2.41E+08	7.97E-08	3470	35349
120	120	2.66E+08	8.41E-08	3871	69080
115	115	2.69E+08	8.56E-08	4026	99555
110	110	2.71E+08	8.47E-08	4150	147757
105	105	2.98E+08	8.72E-08	4656	192552
100	100	3.11E+08	8.64E-08	5034	252139

Un/DoPack (Fine Grain cr8)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev000					
100	98	1.98E+08	7.14E-08	3148	7235
95	95	1.95E+08	7.26E-08	3169	8122
90	90	2.18E+08	7.38E-08	3674	26926
85	85	2.40E+08	7.32E-08	4176	53565
80	80	2.53E+08	7.53E-08	4501	94278
75	75	2.64E+08	7.85E-08	4863	94580
70	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev002					
105	103	2.04E+08	6.79E-08	3157	8224
100	100	1.99E+08	7.32E-08	3178	8908
95	95	1.95E+08	7.02E-08	3178	9700
90	90	1.98E+08	7.02E-08	3336	15622
85	85	2.04E+08	7.17E-08	3532	25189
80	80	2.34E+08	7.77E-08	4213	51456
75	75	2.33E+08	7.86E-08	4270	66071
70	70	2.54E+08	7.60E-08	4762	107035
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
60	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev004					
100	100	1.98E+08	6.87E-08	3148	6723
95	95	1.95E+08	7.12E-08	3169	9072
90	90	1.91E+08	6.98E-08	3216	10913
85	85	1.91E+08	6.97E-08	3298	12060
80	80	2.17E+08	7.59E-08	3847	37634
75	75	2.13E+08	7.24E-08	3903	39072
70	70	2.72E+08	7.85E-08	5151	119123
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev006					
95	95	1.94E+08	7.22E-08	3139	8037
90	90	2.09E+08	7.29E-08	3489	21762
85	85	2.10E+08	7.45E-08	3614	26789
80	80	2.34E+08	7.26E-08	4196	62392
75	75	2.47E+08	7.48E-08	4515	77992
70	70	2.55E+08	7.50E-08	4817	101670
65	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*

Un/DoPack (Fine Grain cr8)					
channel width constraint	routed channel width	area (#min size transistors)	critical path (s)	# of CLBs	run time (s)
Stdev008 / clone					
125	122	2.23E+08	7.15E-08	3151	7172
120	120	2.19E+08	7.11E-08	3199	10904
115	115	2.15E+08	7.33E-08	3223	11154
110	110	2.17E+08	7.29E-08	3320	13852
105	105	2.12E+08	7.30E-08	3314	14911
100	100	2.27E+08	7.33E-08	3611	32670
95	95	2.25E+08	7.37E-08	3715	36062
90	90	2.31E+08	7.68E-08	3899	51642
85	85	2.53E+08	7.62E-08	4387	83331
80	80	2.84E+08	8.01E-08	5056	164058
75	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
70	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev010					
165	162	2.61E+08	7.08E-08	3152	9631
160	159	2.56E+08	7.35E-08	3173	8929
155	153	2.53E+08	7.50E-08	3194	9774
150	149	2.47E+08	7.83E-08	3173	9019
145	144	2.43E+08	7.44E-08	3202	10361
140	139	2.44E+08	7.49E-08	3252	13412
135	134	2.41E+08	7.67E-08	3283	14400
130	130	2.37E+08	7.78E-08	3328	23259
125	124	2.39E+08	7.76E-08	3412	27820
120	120	2.35E+08	7.90E-08	3457	35342
115	115	2.31E+08	7.86E-08	3460	39320
110	110	2.47E+08	7.62E-08	3770	65816
105	105	2.43E+08	8.12E-08	3828	90177
100	100	2.51E+08	8.05E-08	4011	108768
95	95	2.69E+08	7.77E-08	4407	147167
90	90	2.84E+08	8.19E-08	4773	210930
85	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
Stdev012					
155	153	2.52E+08	7.50E-08	3163	10208
150	148	2.47E+08	8.15E-08	3184	10735
145	144	2.44E+08	7.69E-08	3243	20352
140	140	2.45E+08	8.19E-08	3259	16878
135	134	2.49E+08	7.74E-08	3395	26331
130	129	2.53E+08	7.89E-08	3558	40240
125	125	2.47E+08	7.95E-08	3499	41796
120	120	2.50E+08	7.99E-08	3672	68370
115	115	2.54E+08	8.30E-08	3814	112121
110	110	2.80E+08	8.33E-08	4317	172236
105	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*
100	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*	*time exceeded*