INCREMENTAL PLACEMENT FOR FIELD-PROGRAMMABLE GATE ARRAYS

by

David Leong

B.A.Sc., University of British Columbia, 2004

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)

UNIVERSITY OF BRITISH COLUMBIA
November, 2006

Abstract

As the logic capacity of FPGAs continues to increase with deep submicron technology, performing a full recompilation for small iterative changes in a large design is an extremely time-consuming and costly process. To address this issue, this thesis presents a new incremental placement algorithm for FPGAs named "iPlace" that significantly reduces the time required for recompilation. The iPlace algorithm is based on shifting, compaction, and annealing. Key ideas from the algorithm include a placement super-grid that is larger than the physical size of the FPGA. The super-grid allows insertion of additional CLBs into areas with no free locations by CPU-efficient shifting. This is followed by a compaction scheme to re-legalize CLBs that are shifted to illegal locations outside of the physical size of the FPGA. The algorithm ends with a low-temperature anneal to improve quality. This algorithm is capable of handling multiple design changes across large regions of a FPGA. This is especially useful for hierarchical designs where sub-circuits are re-used multiple times. If one such sub-circuit is modified, iPlace can quickly produce a high quality incremental placement solution. For a single region of design change, we found that iPlace is 34 to 260 times faster than the academic tool Versatile Place and Route (VPR) in default mode. Compared to VPR's reduced-quality "-fast" placement option, iPlace is 3 to 28 times faster with equivalent quality. For multiple regions of design changes, iPlace is still 50-70 times faster compared to VPR in default mode when up to 2/3 of the CLBs are modified. Compared to the "-fast" placement option, iPlace is still 5-8 times faster. We believe that iPlace is the first academically available incremental placement algorithm capable of handling significant changes to a netlist for very large circuits.

Table of Contents

List of Tables

List of Figures

Glossary

Application Specific Integrated Circuit (ASIC):	An integrated circuit designed for a specific purpose
Basic Logic Element (BLE):	Basic logic element of an FPGA that consists of a K-input lookup table and a flip-flop.
Computer Aided Design (CAD) Tools:	A set of software tools for designing systems, semiconductor systems in particular
Clustered Logic Blocks (CLB):	Logic Block of an FPGA, comprised of N BLEs clustered together with fast local interconnect
Field Programmable Gate Array (FPGA):	An integrated circuit containing programmable logic elements and programmable routing. An FPGA can be programmed to implement any digital circuit
iPlace:	An incremental placement tool designed for FPGA CAD flows
Interconnect Resource Aware Clustering (iRAC):	FPGA clustering algorithm [25]
Look Up Table (LUT):	A memory cell capable of implementing any K-input, 1 output logic function.
MicroElectronics Corporation of North Carolina (MCNC) Circuits:	A standard set of benchmark circuits used in the FPGA academic community [66]
Minimum Routable Channel Width (MRCW):	The minimum number of routing tracks per channel a FPGA must have to successfully route a particular circuit
T-V Pack:	FPGA clustering algorithm [8]
Un/Do Pack:	An FPGA channel width reduction tool that can target hard channel width constraints to achieve routing closure [1]
Versatile Place and Route (VPR):	An academic placement and routing tool for FPGAs [32]

Acknowledgements

First of all, I would like to thank my graduate supervisor, Dr. Guy Lemieux for having me as his student. Even though I had a full time job aside from this Masters degree, he was patient and supportive throughout the last two years. I am especially grateful for the times Dr. Lemieux was willing to stay late to accommodate my working schedule. Without his brilliant ideas and meticulous screening of my work, I could not have come this far.

I would also like to thank Marvin Tom for getting me started in the field of FPGA research. Without his guidance and help, I would not have been able to quickly learn the intricate details of conducting research. It was a pleasure to work with Marvin on Un/DoPack and the ICCAD paper.

To my best friends, Derek, Patrick and Stephen[1], thank you for the great times we have had since elementary school. My life would be very boring without your company.

Finally, I would like to thank Westgrid for providing the computational resources used to perform the simulations for this thesis. The amount of computation required would not have been possible without the use of the UBC Glacier cluster.[2]

[1] In alphabetical order =)
[2] 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

Dedication

-To my family, thank you for all your support, patience and love during the last few years.

1. Introduction

Field Programmable Gate Arrays (FPGAs) are a specialized type custom integrated circuit. FPGAs are capable of implementing any digital circuitry through the use of reprogrammable look up tables and routing fabric. Compared to ASICs, FPGAs offer a low cost alterative to designing digital circuitry. FPGAs allow designers to quickly prototype and test a circuit on hardware, without the need for expensive mask and foundry costs associated with ASIC designs. However, there are drawbacks of using FPGAs compared to ASICs. ASICs offer much higher chip density, faster clock speeds as well as lower power usage. The tradeoffs associated with ASICs and FPGAs depend on the sales volume of the final product. ASICs are more suited to large volume, high performance devices such as microprocessors. FPGAs on the other hand are more suitable for medium-to-small volume devices or devices that require fast time to market.

As the logic capacity of FPGAs increases with deep submicron technology, the run time required to compile and fit a high level design onto a target FPGA increases significantly. Recompiling the design from scratch for small changes or localized improvements is a time-consuming operation. For today's largest FPGAs, a full recompilation often requires several hours to execute the entire FPGA CAD flow.

Often only a small section of the circuit is modified, thus raising the need for incremental algorithms that can speed up the compilation process. The need for incremental algorithms extends beyond small design changes. For circuits designed using design hierarchy, multiple sub-circuits can be modified concurrently by several designers. The need for incremental CAD techniques that can quickly integrate multiple changes together are necessary for fast turn around time. In addition, sub-circuits can be re-used multiple times in a design. Modifications to one such sub-circuit require changes to multiple regions across the FPGA. Incremental techniques must be able to handle the use cases outlined above. In addition, incremental techniques must be scalable for small-to-large modifications to ensure the most efficient use of development time. The resulting incremental compilation for such use cases should be as high quality as compilation from scratch.

1.1. Contributions

This thesis presents an incremental placement algorithm named "iPlace" to be used as part of an incremental CAD flow. Our approach uses existing academic tools as a foundation and extends them for incremental placement. Results from this work appeared in [1]. A paper describing the algorithm details has been submitted for review [2].

The iPlace algorithm starts with an initial high-quality placement of a "before" circuit prior to modification. For this, we use the default VPR placement algorithm. The "before" state is used as a reference solution for incremental placement. Next, iPlace finds a new placement for the modified "after" circuit state. The "after" state is the new circuit to be incrementally placed using iPlace. The iPlace algorithm produces the new placement solution through four phases: (1) use the "before" state to produce an initial placement of the unmodified CLBs, (2) insert modified CLBs by shifting into a *super-grid*, (3) compact the super-grid by shifting to re-legalize placement, and (4) refine with a low-temperature anneal. The *super-grid* is a placement grid that is extended beyond the bounds of the initial FPGA device.

Three key ideas are present in the iPlace algorithm. The first idea is the use of floor-planning constructs to constrain the initial placement of modified CLBs close to the original placement "hole" left by the removal of some CLBs from the "before" state. These floor-planning constraints are continuously modified as the algorithm executes. The second idea is a "placement super-grid". The super-grid expands the entire placement region as more CLBs are added than the "hole" permits. The third idea is the use of partial design shifting during the expansion and compaction phases of the algorithm. Shifting partial rows/columns of CLBs is much more CPU-efficient than relying upon individual moves used in annealing-based approaches, particularly because annealing must also measure the cost of each LE move before deciding. After the shifting, a low temperature simulated annealing run is performed to improve the solution quality. This annealing step is the slowest step. These steps combine to produce a fast and high quality incremental placement algorithm

To measure the performance of the incremental placement algorithm, we developed three sets of benchmarks. Each individual benchmark circuit has a "before" and "after" state. The first set of benchmarks approximate incremental design changes where subregions of a circuit are significantly modified in logic and structure but other regions are untouched. This set is created using a synthetic circuit generator to mutate a selected subregion of the circuit. These clones are re-stitched into the original circuit [3] [4]. The second benchmark set approximates a physical resynthesis flow where the circuit is logically identical, but functionally-equivalent changes are made within one local region. The second benchmark set is automatically produced from a physical resynthesis CAD flow [1]. The third benchmark set is a variation of the second, but scaled to include multiple regions of design changes. This set, also produced using [1] is used to determine the ability of iPlace in handling multiple design changes across large regions of the chip.

1.2. Thesis Outline

The remainder of this thesis is organized as follows. First, Chapter 2 will present background information on FPGAs and prior work related to incremental placement. Next, Chapter 3 will describe the iPlace algorithm in detail. Chapter 4 will present the results of benchmarking iPlace with the three benchmark sets. Chapter 5 presents a qualitative comparison of iPlace against existing fast and incremental approaches. Finally, Chapter 6 summarizes the key contributes and results of iPlace.

2. Background

This chapter presents the background information related to FPGAs and incremental placement research. First, the hardware architecture and makeup of FPGAs will be presented. Second, the software CAD flow related to compiling designs for FPGAs will be discussed. Finally a summary of prior work related to incremental placement will be presented.

2.1. FPGA Architecture

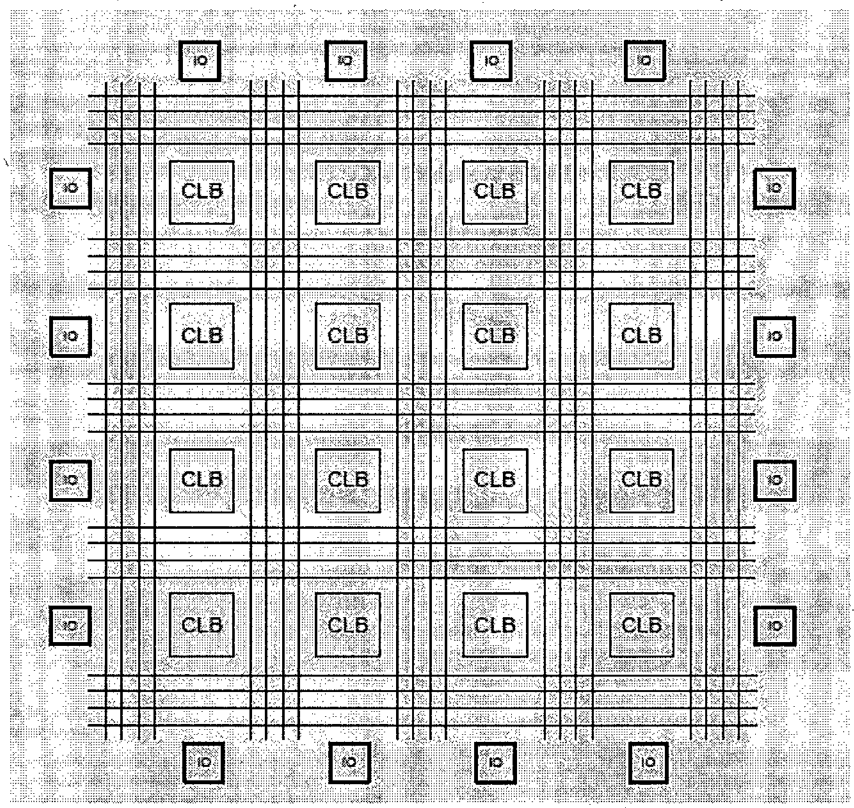

Figure 1 Island Style FPGA [5]

Field Programmable Gate Arrays (FPGAs) are Very Large Scale Integrated (VLSI) circuits that are capable of implementing *any* user designed digital circuit. FPGAs offer this flexibility of through the use of reconfigurable Logic Elements (LE). The typical layout of an FPGA circuit used by commercial devices such as Altera's Stratix [6] and Xilinx's Virtix [7] is shown in Figure 1. The layout style of the FPGA shown in Figure 1 is called an Island Style Architecture. The Island Style FPGA consists of clusters of LEs called Configurable Logic Blocks (CLB). The CLBs are laid out in a rectangular array surrounded by configurable routing wires in both the vertical and horizontal directions. The *Array Size* of an FPGA is measured in terms of the number of CLBs spanning horizontally and vertically. The name "Island" architecture comes from the fact that CLBs "islands" are surrounded by a sea of configurable wires. Around the periphery of the chip are Input/Output (IO) pads to connect the FPGA to the rest a circuit design.

Logic Element and Clustered Logic Blocks (LE and CLB)

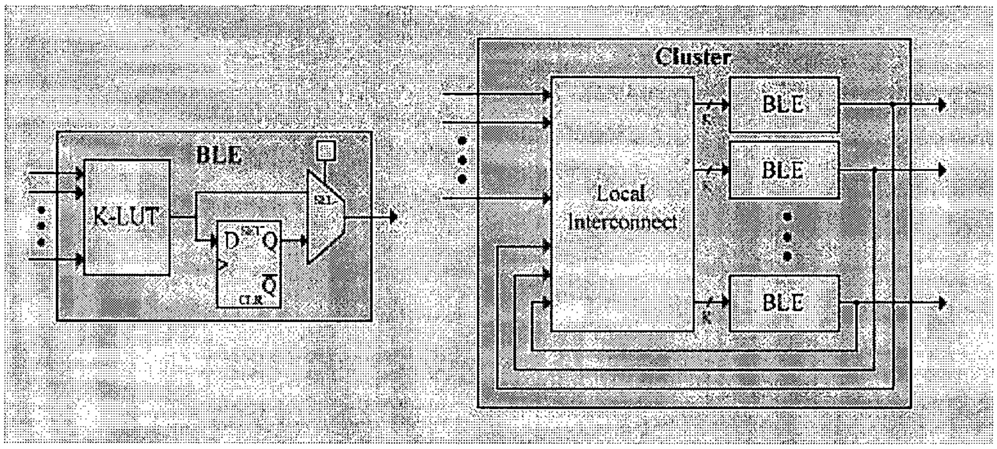

Figure 2 BLE and CLB [5]

The most basic element of a FPGA is the Basic Logic Element (BLE or LE). The logic element consists of a K-input look up table, a flip-flop and a multiplexer (Figure 2). The k-input LUT is capable of implementing any k-input, 1 output combinational logic function. A LUT is made of 2^K configuration bits that can be programmed to implement the desired logic function. The flip-flop provides synchronous output for the logic function implemented by the LUT. Finally, the multiplexer allows the selection of the combinational or synchronous output for the BLE.

From prior research [8], it was shown that it is more efficient to pack multiple LEs into a *cluster* called a Configurable Logic Block (CLB). The structure of a CLB is shown in Figure 2. Each cluster contains N LEs. The number N is typically 4[3], 8[4], 10[5] or 16[6] logic elements [6][7]. Each LE within the cluster is interconnected to each other with fast local interconnect. The number of inputs to a CLB I, is smaller than the maximum K*N because LEs can share common inputs.The advantages of packing multiple LEs into a cluster include reducing delay, reducing the amount of interconnect required, more dense FPGA designs and improved CAD compilation runtime.

[3] Xilinx Virtex
[4] Xilinx Virtex II
[5] Altera Cyclone, Altera Stratix
[6] Altera Cyclone II

Routing Architecture

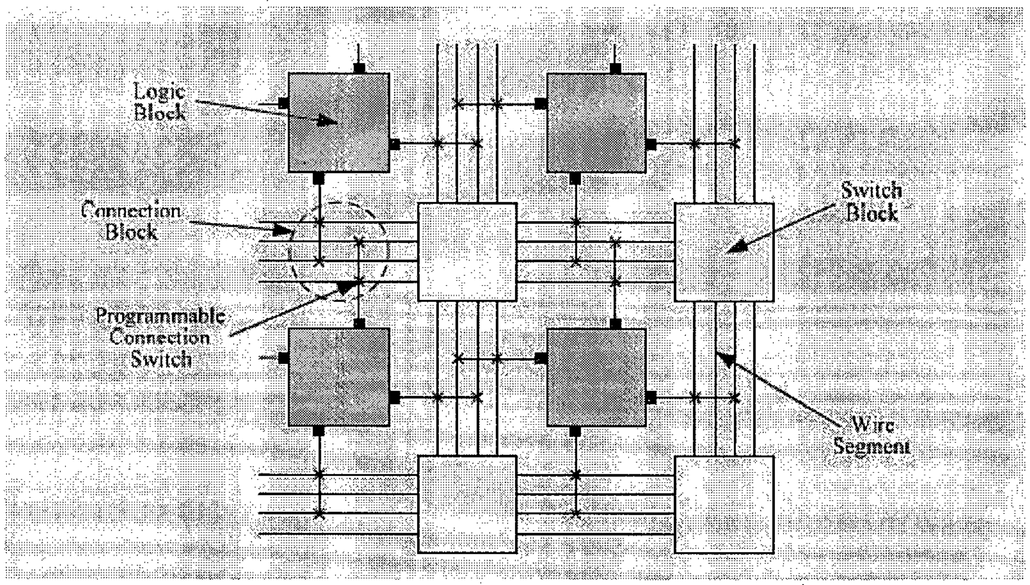

Figure 3 FPGA Routing Architecture [5]

The second component that makes up an FPGA is the routing architecture. The routing architecture is responsible for connecting the reconfigurable CLBs together to form the overall design. The routing architecture is comprised of three components: wires, switch blocks and connection blocks [9]. A simplified view of the routing architecture is shown in Figure 3. First, wires are the core of the routing structure. Wires fill the vertical and horizontal channels in between rows/columns of CLBs. In commercial FPGAs, multiple length wires spanning 4, 8, 16 and 24 or greater number of CLBs [6] are provided for the different routing requirements of each net. Shorter wires are for local interconnection, whereas longer wires are designed to connect CLBs far apart with less delay. Wires can be can designed to carry signals bi-directionally or uni-directionally, although the latter has been shown to be faster and consume less area in recent research [10]. The second component is the switch block. The switch block connects vertical and horizontal wires so that signals can switch directions or extend the length of a wire past the length of a segment. The third component is the connection block. The connection block connects CLBs to their adjacent wires. In bidirectional interconnect, the switch and connection blocks are often fabricated using tri-state buffers or pass transistors, and controlled using sram programming bits. In unidirection interconnect, the switch and connection blocks are fabricated with multiplexers, tri-state buffers and controlled using sram. The routing architecture is the predominant contributor to the FPGA die size, speed and power requirements.

Commercial devices:

Part	EP1S10	EP1S20	EP1S25	EP1S30	EP1S40	EP1S60	EP1S80
Logic Capacity	10,570	18,460	25,660	32,470	41,250	57,120	79,040
Channel Width	160	160	160	160	160	160	160

Table 1 Altera Stratix Family of FPGAs [6]

Commercial devices measure the logic *capacity* of a FPGA in terms of the number of the number of LEs. The routing *capacity* is measured in terms of the number of routable wire segments passing between adjacent CLBs. This fixed routing capacity is called the *Channel Width* of a device. Commercial devices also contain multiplier and memory blocks that are not within the scope of this work. Commercial devices are usually offered as "families" of FPGAs, where the channel width remains the same within a family, but the array size is increased to offer larger logic capacities for different family members. An example of an FPGA family from Altera is presented in Table 1. The latest FPGAs offered by Altera and Xilinx have logic capacities exceeding 200,000 LEs. [6][7][11]

2.2. FPGA CAD Flow

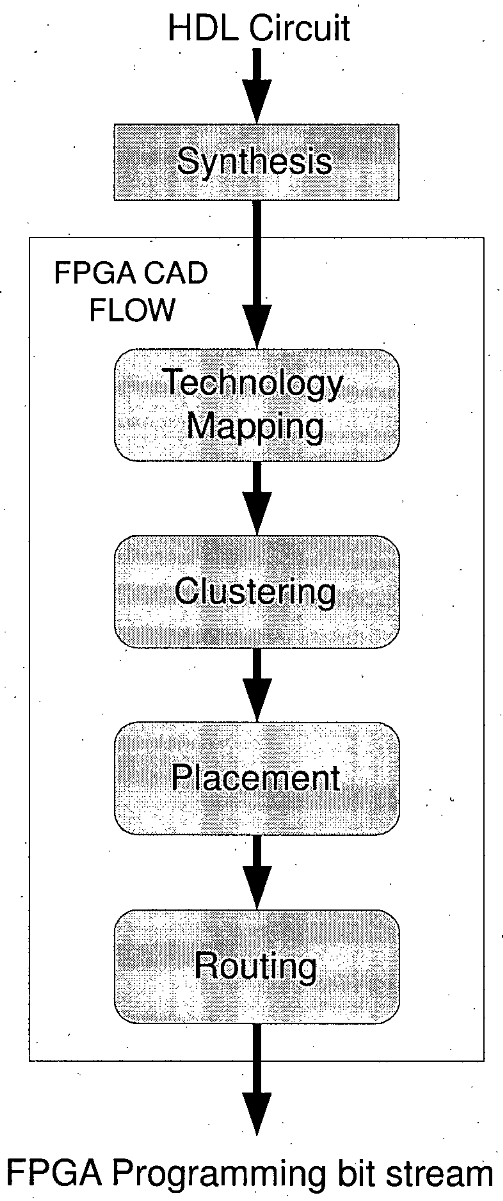

Figure 4 FPGA CAD Flow

To develop a circuit for use on a FPGA, designers use high level languages such as VHDL or Verilog to describe the circuit behavior. A multi-stepped CAD flow is required to compile the circuit description from the high level language description to programming the LUTs and switches of the FPGA. This CAD flow consists of five steps. The first step is synthesis. In this step, the circuit description is compiled from the high level language to a network of Boolean equations and flip-flops. The synthesis step is common to both ASIC and FPGA development. The FPGA specific portion of the CAD flow consists of technology mapping, clustering, placement and finally routing. In this section, each step of the FPGA CAD flow will be presented.

2.2.1. Technology Mapping

The first step in the FPGA CAD flow is technology mapping. In this step, the Boolean network is mapped to the look-up table size of the FPGA. The goal of technology mapping is to use as few LEs as possible to minimize logic usage and circuit delay. Technology mapping algorithms minimize the delay of a circuit by minimizing the *logic depth*, which is the longest path of a circuit. Several technology mapping algorithms are presented in [12][13][14][15][17][18][19][20][21].

The most notable technology mapping algorithm for FPGA is FlowMap [12][13]. The FlowMap algorithm was revolutionary as the algorithm is able to produce a depth optimal solution in polynomial complexity time. This algorithm is based on network flow algorithms using the *max-flow-min-cut* theorem. The results reported in [12] shows that FlowMap produces better critical path and area results compared to other technology mapping algorithms at the time. Follow-ups to the FlowMap algorithm include CutMap [14][15] and DAOMap [16] . CutMap improved on FlowMap by reducing logic duplication. Logic duplication occurs during the FlowMap algorithm when a node is encapsulated (duplicated) by multiple LUTs in the technology mapping solution. CutMap is able to reduce the number of logic elements required by 20% compared to FlowMap while maintaining depth-optimality in the solution.

2.2.2. Clustering

The second step in the FPGA CAD flow is clustering. In this step, the technology mapped K-input LUTs are packed clusters of size N defined by the FPGA architecture. The goal of clustering is to maximize utilization of the cluster resources, minimize delay and to reduce the amount of interconnect required between clusters. Clustering algorithms can be classified into three categories. The most common algorithms are bottom-up [8][22][23][24][25]. Bottom up algorithms greedily builds each cluster by selecting a seed LE and growing by individual LEs. The second type of clustering algorithm is top-down [26][27]. Top-down algorithms start with the entire technology mapped circuit, and recursively partitions the circuit into bins until each bin reaches the cluster size. The final type of clustering algorithm is depth optimal clustering [28][29][30]. Depth optimal clustering minimizes delay in exchange for more logic use due to duplication.

Bottom up algorithms are the most common category employed for FPGAs, so the bottom up approach will be discussed in this section. Bottom up algorithms begin each cluster by selecting a seed LE based on an algorithm specific criterion. The algorithm then iteratively selects the next most attractive LE using an attraction function to add to the cluster. Once a cluster is full, a new seed is chosen for a new cluster. The process is repeated again until all LEs are clustered.

$$Criticality(b) = 1 - \frac{\text{slack(b)}}{\text{max\_slack}} \quad (1)$$

$$Attraction(b,c) = \alpha * crit(c) + (1-\alpha) * \frac{nets(b) \cap nets(c)}{G} \quad (2)$$

A widely cited academic FPGA bottom-up clustering algorithm is T-VPack [8]. T-VPack is a timing aware algorithm that aims to minimize both the delay and the number of nets to route. The seed for each cluster is chosen as the most timing critical LE. The equation to calculate the criticality of an LE "b" is given in (1). Additional LEs are added to the cluster using the attraction function shown in (2). Equation (2) shows that the attraction of an element c to cluster b is a factor combining the criticality of the LE and the normalized[7] number of nets common with the current cluster contents. The weighting factor α balances the importance of timing versus the number of nets absorbed into the cluster. The factor α was experimentally determined in [8] to be 0.75.

Another bottom-up clustering algorithm is iRAC [25]. iRAC is a greedy clustering algorithm that aims to minimize the number of nets to route for the computed clustering solution. The premise of the algorithm is to encapsulate as many nets as possible within a cluster, thus minimizing the number of nets required to route between clusters. iRAC also includes a cluster input limiting component computed based on the Rent parameter [31] of the cluster. The clustering algorithm used for this work is an iRAC replica produced for [1]. The iRAC replica excluded the input limiting component due to the limited improvement it offered. This algorithm was selected for use with this thesis because it produces the lowest routed channel widths and delay performance compared to TVPack [25].

2.2.3. Placement

The third step in the FPGA CAD flow is placement. In this step, the clustered LEs are mapped to physical CLB locations on the FPGA. The goal of placement is to minimize both the routing delays of regular nets and the delay of the critical path. The placement engine must also be congestion aware to avoid over-usage of the routing resource, which

[7] The number of nets in common is normalized to a constant G

could lead to an un-routable solution. There are three general types of placement algorithms. First, Simulated Annealing (SA) algorithms [5][32][33][34][35] are the most common and based on a hill climbing approach. Second, Analytical Placement algorithms [36][37][38][39] solve the placement problem by using systems of equations. Analytical Placement algorithms are good at finding approximate locations for CLBs, but due to the discretization effects of mapping to an FPGA array, they often result in overlaps that need to be removed through additional re-legalization steps. Finally, Min-Cut algorithms [40][41][42] produce placement solutions using partitioning techniques. Simulated Annealing algorithms are the most common in academic research and will be presented in more detail in this section.

```
Random_Placement();
temp = Initial_Temperature();
$R_{limit}$ = nx;

While( ExitConditionSatisfied() == false )
  for( int i=0; i<inner_num; ++I )
    move = Generate_Move();
    $\Delta C$ = Calculate_Cost_Delta( move );
    float r = rand(0,1);
    if( $\Delta C < 0$ )
      AcceptMove( move );
    else if( $r < e^{-\Delta C/T}$)
      AcceptMove( move );
  end for
    temp = UpdateTemperature(); // $t = \alpha * t$
    $r_{limit}$ = UpdateRange();
    Update_Net_Criticality();
end while
```

Figure 5 Simulated Annealing Algorithm

VPR is the academic FPGA CAD tool most commonly used for FPGA research. The work presented in this thesis is developed by extending the VPR simulated annealing placement algorithm for incremental placement. VPR features an adaptive temperature cooling schedule and range limiting techniques to produce excellent placement solutions.

The VPR simulated annealing algorithm is presented in Figure 5. The first step in the placement process is to randomly place all CLBs onto the placement array. Next, the initial temperature is determined by performing a number of test swaps equal to the number of CLBs. The initial temperature for VPR is set as 20 times the standard deviation of the cost evaluated during the test swaps. The initial range limit is set as the length or width of the device[8].

$$\Delta C = \lambda \frac{\Delta \text{Timing\_Cost}}{\text{Previous\_Timing\_Cost}} + (1-\lambda)\frac{\Delta \text{BB\_Cost}}{\text{Previous\_BB\_Cost}} \quad (3)$$

$$\text{timing\_cost} = \sum_{\forall i,j \in circuit} delay(i,j) * crit(i,j)^{CE} \quad (4)$$

$$\text{bb\_cost} = \sum_{i=1}^{\forall nets} q(i)[bb_x(i) + bb_y(i)] \quad (5)$$

The simulated annealing algorithm works by performing *inner_num* number of swaps at each temperature setting. For each swap, a random move is proposed by swapping the placement position of two CLBs. If the swap results in a reduction in the placement cost, the swap is accepted. If swap results in an increase in cost, a probabilistic acceptance is used based on the $e^{-\Delta C/T}$ where ΔC is the change in cost according to (3) and T is the current temperature. Accepting bad moves is required for hill climbing to avoid local minimums in the cost function. Equation (3) is composed of 2 terms, a timing cost and a bounding box cost. The factor λ is a weighting factor to balance the importance of these costs. Each term

[8] Assuming a square array. Commercial FPGAs are rectangular.

is normalized to the total cost from the previous iteration. The timing cost shown in equation (4) is a sum of the delay*net criticality product for each net. The net criticality is defined as (1). The bounding box cost is the sum of the Manhattan distance for each net.

After *inner_num* number swaps are performed at each temperature range, the temperature and range limit factors are updated. The temperature is degraded using an update factor α. As the temperature is lowered during the annealing process, the $e^{-\Delta C/T}$ term approaches 0 so that fewer bad swaps are accepted for hill climbing. Both α and the range limit are adaptively adjusted to keep the swap acceptance rate at ~44%. The 44% acceptance rate was shown to be the optimal hill climbing factor in [43].

$$t < \frac{0.005 * cur\_cost}{num\_nets} \quad (6)$$

The algorithm terminates when the exit criterion is met. The equation for calculating the exit criteria in VPR is presented in (6)

2.2.4. Routing

The final step in the FPGA CAD flow is routing. In this step, routing resources are assigned for each net of the circuit to connect the logic elements together based on the CLB placement solution from the placement stage. The goal of routing is to minimize the delay of the circuit and to avoid congestion in the routing resources. Routing algorithms can be classified into two categories. First, two-step routers [44][45][46][47][48] break down the routing process into two steps: global routing and detail routing. Second, single step global-detail routers [49][50][51][52] perform both the global and detail routing together as a single step. In global routing, the input-output pins and the routing channels are assigned to a signal. In detail routing, each signal is assigned to specific routing track within each channel. Two-step routers are used for ASIC designs because of routing flexibilities. Single step global-detail routers are used for FPGAs due to the limited and constrained routing resource architecture.

The academic CAD tool VPR [5][32] features a two step FPGA router using the PathFinder routing algorithm [50]. PathFinder is an iterative routing algorithm based on cost sharing and track negotiations. During the initial routing iteration, all nets are routed via the lowest cost route. This produces overuse in popular tracks where multiple signals are assigned to the same track, which is illegal for a routing solution. For each iteration, all nets are ripped up and rerouted. Iteratively, the cost of the overused wire segments is increased so that nets are forced to evaluate and use different routes wherever possible. The algorithm terminates when there is no more overuse, forming a legal solution.

$$Cost(i) = Crit(i) * delay(i) + (1 - Crit(i)) * b(i) * h(i) * p(i) \quad (7)$$

The VPR routing cost function is shown in (7). The cost function to route each net is based on a combination of two terms. The first term represents the Elmore delay of the wire segment. The second term is based on the base cost *b(n)*, history cost *h(n)* and the present cost *p(n)*. The history cost represents a history of how congested a wire segment has been. The present cost represents the present cost to use a wire segment. The present cost is increased as the algorithm progresses to discourage over usage of wires. The weighting of each term is dependent on the criticality of the net calculated using (1). Nets on the critical path will have a criticality factor close to 1. Based on (7), the critical path net will be routed on the path with lowest delay.

2.3. Incremental Placement Techniques

In a traditional FPGA "full compilation" process, the entire CAD flow must be executed if any changes are made to a circuit. To speed up this process, our flow takes an *incremental* approach: only the *changes* to the netlist are propagated through clustering and placement, and a full route is done at the end. For both incremental clustering and incremental placement, a "reference solution" computed from the previous compilation is used to identify changes and reduce the amount of new work. This section will briefly describe both of these steps, but the focus is on incremental placement.

Incremental clustering initially starts with the previous list of CLBs, a list of unmodified CLBs, and a list of unclustered logic elements. The unmodified CLBs are the same as before (they contain the exact same logic elements). In contrast, modified CLBs arise because some logic elements were *deleted* by a user logic change. Modified CLBs are unclustered into its constituent LEs, and these are added to the pool of new LEs that were *added* by the user logic change.

Incremental clustering proceeds as follows. The unmodified CLBs are kept as-is. Due to their greedy nature, clustering algorithms such as TVPack [8] and iRAC [25] can easily treat these unmodified CLBs as an intermediate solution and continue grouping the unclustered logic elements into new CLBs. Since all flip-flop locations in the entire circuit are known, incremental clustering can still identify critical paths and remain timing-driven. Our implementation uses the iRAC replica, since it produces good timing results and requires the lowest channel width for routing. The clustering tool then proceeds to form new CLBs using the new LEs and LEs from unmodified CLBs

Incremental placement initially starts with the previous placement of CLBs, a list of unmodified CLBs, a list of removed CLBs, and a list of new CLBs. In the event of multiple changes[9], the list of new CLBs (and removed CLBs) is divided into a number of sub-lists, one for each change or each instance. Optionally, a rectangular floorplan constraint for each change can be given. If none is provided, the bounding box for each "removed CLB sub-list" is computed and applied as a floorplan constraint to the corresponding "new CLB sub-list".

Incremental placement proceeds as follows. The unmodified CLBs are re-placed in their previous location to preserve "spatial locality", *i.e.*, physical closeness to their connected neighbours. For each "region" of change, the placement locations previously occupied by the removed CLBs are now left empty, thus leaving white space for the new CLBs. For the new CLBs, two cases must be considered. If there are fewer new CLBs, they all fit in the white space left behind and placement is "trivial". The second case to consider is when the new CLBs exceed the removed CLBs. Since there is insufficient room, unmodified CLBs *must* be moved to make room or new CLBs will be placed far away. This is the key problem to solve for incremental placement. To maintain placement locality, most incremental placement algorithms temporarily allow overlap, which is when multiple CLBs occupy the same physical location. Overlap results in an illegal solution which must be fixed through a lengthy *re-legalization* step. Instead, iPlace immediately shifts other CLBs out of the way and continues. This is called expansion. By the end, CLBs are often pushed past of the limits of the array, which is also illegal. iPlace then re-legalizes by compacting: it forcibly shifts available whitespace along the edges to where it is needed, thus making room for the illegal CLBs. Finally, this legal placement is refined through annealing.

[9] Due to component re-use, multiple instances of the same circuit are treated as multiple changes.

Incremental placement algorithms have not been widely published. Cong and Sarrafzadeh [53] give high level overviews of the problems associated with incremental CAD, including placement. They note two separate needs for incremental placement: to optimize an existing good placement for a new metric, or for handling the addition and removal of logic or nets. iPlace is designed for the latter situation.

Previously published algorithms for incremental placement in FPGAs include work by Singh and Brown [54], Suaris et al [55] and Togawa et al [56]. Both [54] and [55] are intended to be used with physical resynthesis to assist with timing closure, while [56] is a congestion- driven approach. A comparative summary follows.

The Singh and Brown placer, ICP [54], is primarily focused on improving timing through small netlist changes, such as retiming register moves. As a result, it operates on changes at the logic element (LE) level. Initially, ICP allows all LEs to be assigned to their preferred location, which may be illegal, for best timing performance. Then, it iteratively moves each LE, using a negotiation similar to PathFinder, to legalize conflicts and reduce timing and estimated wiring costs. It considers individual LE moves, and evaluates the cost of each one. Since it considers only a small number of moves, it is about 8 times faster than VPR. In contrast, iPlace is about 60 times faster than VPR because it operates at the CLB level and does not evaluate individual move costs when resolving illegal placements. Instead, iPlace presumes that shifting imposes a minimal cost penalty during legalization and uses simulated annealing at the end to improve or recover lost quality.

Suaris et al present an incremental placer in a framework called IPR [55] which has very similar goals and operation to ICP. However, IPR uses quadratic placement to assign initial placements, which also results in overlaps. Like ICP, IPR also performs individual LE moves and evaluates the cost of each one during legalization – however, the IPR costs appear to be entirely timing-driven. No run-time results were reported for IPR; hence it cannot be directly compared to iPlace.

Togawa et al present a congestion-driven incremental placer [56] that shifts overlapping individual LEs to reduce global routing capacity. It avoids increases in channel width for up to 20% changed LEs. In contrast, we have observed that iPlace can tolerate changes to 2/3 of all CLBs without inflating channel width, critical path or wirelength.

Previous work on incremental placement for standard cells includes papers by Choy et al [57], J. Li et al [58], and Z. Li et al [59]. The two algorithms in [57] insert one cell at a time by computing the most desired location and the lowest-cost shift sequence of the nearest empty cell. The approach in [58] eliminates overlap by shifting entire floorplan rectangles; however, it assumes significant whitespace gaps between rectangles. The approach in [59] inserts one cell at a time into optimal position, and each time it legalizes by solving an integer programming problem that determines how to shift the fewest cells the least distance. Both [57] and [59] are meant for very small netlist changes and would likely be too slow for use within iPlace. If an original floorplan is available, [58] would be useful. However, iPlace does not presume any floorplanning – it constructs its own floorplan constraints using information from the changed elements.

The final type of incremental placement algorithms are commercial tools such as Altera's Quartus II and Xilinx's ISE. These tools also support incremental compilation. The tools have an advertised speedup of 2-3 times for the entire FPGA CAD flow when comparing an incremental compilation versus a full compilation. The details of the algorithms employed by the commercial tools are proprietary and not known in detail.

The approaches taken by previous incremental algorithms all involve solving the problem of overlaps. The algorithms start with an initial best but illegal placement, then iteratively resolves the illegal locations using different schemes until a valid placement is produced. In comparison, iPlace approaches the overlap problem in a novel manner. Instead of allowing overlaps in the first place, a CPU efficient shifting scheme is used to shift entire rows or columns of CLBs to create more white space for insertion. The shifting is followed by a fast and tuned simulated annealing (VPR) run for optimization.

2.4. Fast Placement Techniques

In addition to incremental placement techniques, fast placement techniques can also be considered for incremental placement. Fast placement techniques sacrifice quality in exchange for faster run time. Several fast placement techniques by Hauck et al. [60], Sankar et al. [61] and Tessier [62] evaluate different algorithms and tradeoffs for faster placement.

Hauck et al. [60] presents several fast placement techniques including partitioning, force-directed and simulated annealing algorithms. The techniques were evaluated by their run time versus critical path quality tradeoff. The best quality results were achieved with simulated annealing (VPR). To reduce run-time, the *inner_num* parameter was varied to reduce the number of swaps. By reducing the run-time, it was found that a 20 times speed up resulted in a 2x increase in critical path. It was also found that a 2.5 times speedup had a 1.34x increase in the critical path. Force-directed placement techniques had similar run-time trade offs as simulated annealing. In comparison to iPlace, iPlace is capable of 60x speedup with no quality degradation. The approach of reducing the *inner_num* parameter to reduce run-time is common between [60] and iPlace.

Sankar et al. [61] presents a fast placement technique based on multi-level clustering and fast simulated annealing refinement. The algorithm performs recursive bottom up clustering to form clusters of CLBs. After each stage of clustering, each cluster is internally placed constructively and refined using a retuned simulated annealing algorithm (VPR). Although only wire-length results were presented, the tool produced better quality results compared to VPR (non-timing driving) in "-fast" mode with similar run time.

Tessier [62] presents the Frontier fast placement system used in conjunction with pre-fabricated macro blocks. The Frontier approach is similar to Sankar et al. where clustering is used to group CLBs [macro blocks for Frontier] prior to simulated annealing refinement. The Frontier system was designed to place groups of pre-placed CLBs as macro blocks. The Frontier system has a reported 17x speedup compared to commercial Xilinx software.

In summary, fast placement techniques employ a variety of different methods to speed up the run-time. One common attribute to all three algorithms is the use of a retuned, fast simulated annealing (VPR) refinement scheme to improve quality. This approach is also adopted by iPlace to refine the incremental placement solution after the expansion and compaction phases. The run time of these fast algorithms are comparable to iPlace. The main difference is the quality of the placements produced. iPlace is able to produce placement solutions as *high* quality as complete placement from scratch. In comparison, the fast placers trade the decreased runtime with significantly reduced quality.

3. iPlace Algorithm

This chapter provides an in depth explanation of the iPlace algorithm. The iPlace algorithm is a 4 step approach to incremental placement. The core idea of the placer is based on spatial locality. If an element was previously placed at a particular location, then it is very likely that it should be placed at the exact same location (or nearby) after the circuit has been modified. A second paradigm employed by iPlace is *simplicity.* iPlace avoids the use of heavy computation for the first three phases, and only uses limited annealing to cleanup the final solution. The limited use of annealing or other computational intensive algorithms is key to iPlace's performance. The four phases of iPlace are as follows:

1. Initial Placement and Floorplanning
2. Super-grid Expansion Placement
3. Compaction (Re-legalization)
4. Refinement by Low Temperature Annealing

The iPlace algorithm is implemented in the VPR framework. Three inputs are required for the incremental placement process. The first input is an initial placement from the "before" circuit state. The second input is a *floorplan* or rectangular region identifying approximately where to place the changed elements. The third input is the modified or "after" circuit state. iPlace identifies which CLBs are *modified* and which are *unmodified* by comparing the first and third input data.

3.1. Initial placement

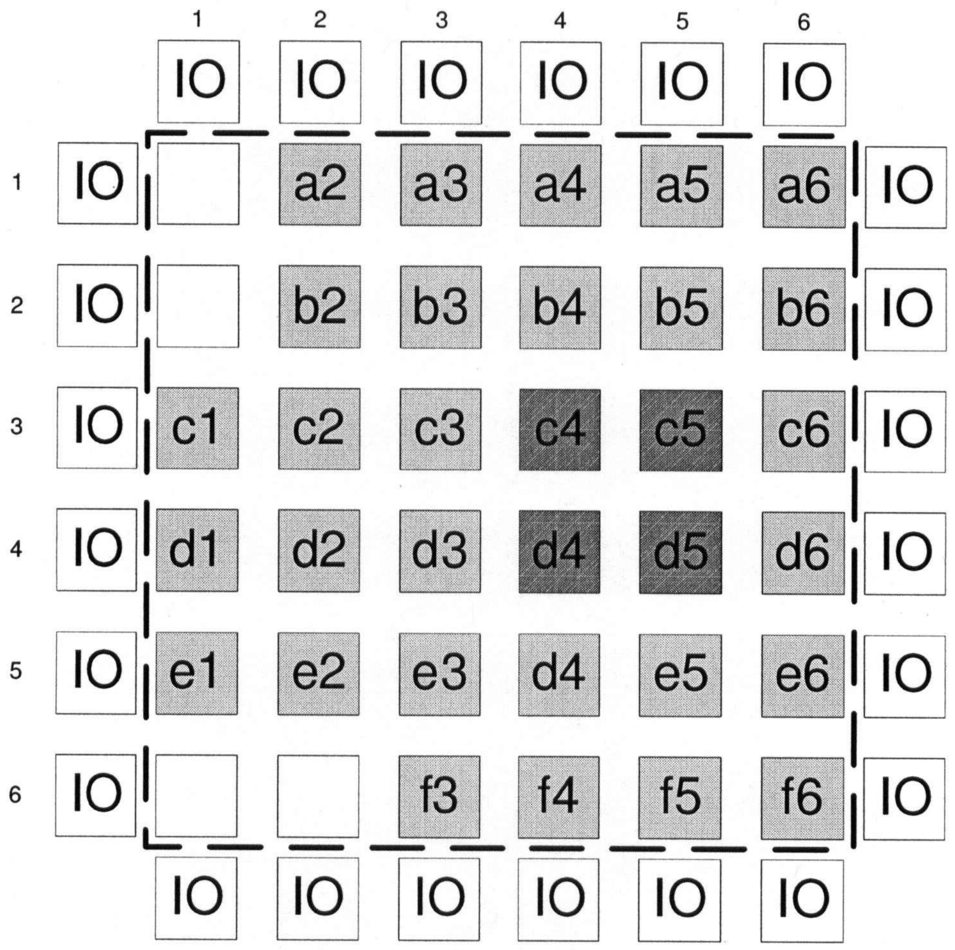

Figure 6 Initial Layout

The first phase of iPlace is to provide an initial placement for all *unmodified* CLBs by examining the placement solution of the "before" circuit state. This step is pictorially shown in Figure 6. The labelled cells represent unmodified CLBs; these are initially placed in their previous placement locations to maintain spatial locality. The hashed cells represent CLBs that have been modified. These are removed from the initial placement, leaving holes to be filled in later by the *modified* CLBs.

3.2. Floor-planning

The holes left behind by the removed elements are also the basis for floor planning. The argument is that any modified CLBs should be placed where the holes were created to preserve spatial locality. iPlace is actually capable of handling *multiple modification regions*. For each hole left behind by a group of modified CLBs, a floorplan rectangle can be generated to guide the replacement CLBs into that specific region. For the example in Figure 6, a floorplan rectangle is generated at location (4, 3), with a size of 2x2.

In this thesis, we are not overly concerned with the precise method of identifying a floorplan region as part of the incremental placement algorithm. Floorplans can also be constructed with the following methods. First, CAD tools already allow designers to floorplan the usage of a device. These constraints can be translated into incremental floorplan regions based on the modifications made on the circuit. The use of design hierarchies and SOC methods can also be used to create floorplan regions. The placement region of each component in the hierarchy can be used to dictate the region specified for incremental changes. Finally, incremental placement required for iterative re-synthesis CAD flows are also supported by iPlace. Algorithms that target constraints such as the most congested regions can directly translate the re-synthesized areas into floorplan constraints.

3.3. Expansion

```
initial_placement()
shift = 0;
for each floorplan region f
    {
    for each modified CLB c of floorplan f
        {
        if num free space is 0
            {
            shift%4 == 0 ? shift right by 1
            shift%4 == 1 ? shift up by 1
            shift%4 == 2 ? shift left by 1
            shift%4 == 3 ? shift down by 1
            shift++
            expand_affected_floorplans()
            }
        randomly place CLB c in floorplanned free space
        }
    }
place_any_un_floorplanned_clbs()
```

Figure 7 Super-grid Expansion Pseudocode

The second phase of iPlace is the insertion of the *modified* CLBs into the placement grid. Each modified CLB is associated with a floorplan region. The floorplan is used give a rough initial location or area where the modified CLB should be placed. The number of modified CLBs could exceed the number of free spaces available in the floorplan area. In the expansion phase, a CPU efficient shifting scheme is used to overcome the limitation of insufficient placement locations. This phase is called "expansion" because the shifting allows CLBs to be shifted *outside* of the normal placement area. This increases or *expands* the placement grid to create more room. We call the result a "super-grid", which includes the original placement area and all of the outside areas. Precise pseudocode for this step is shown in Figure 7. The algorithm will be explained as follows using an example.

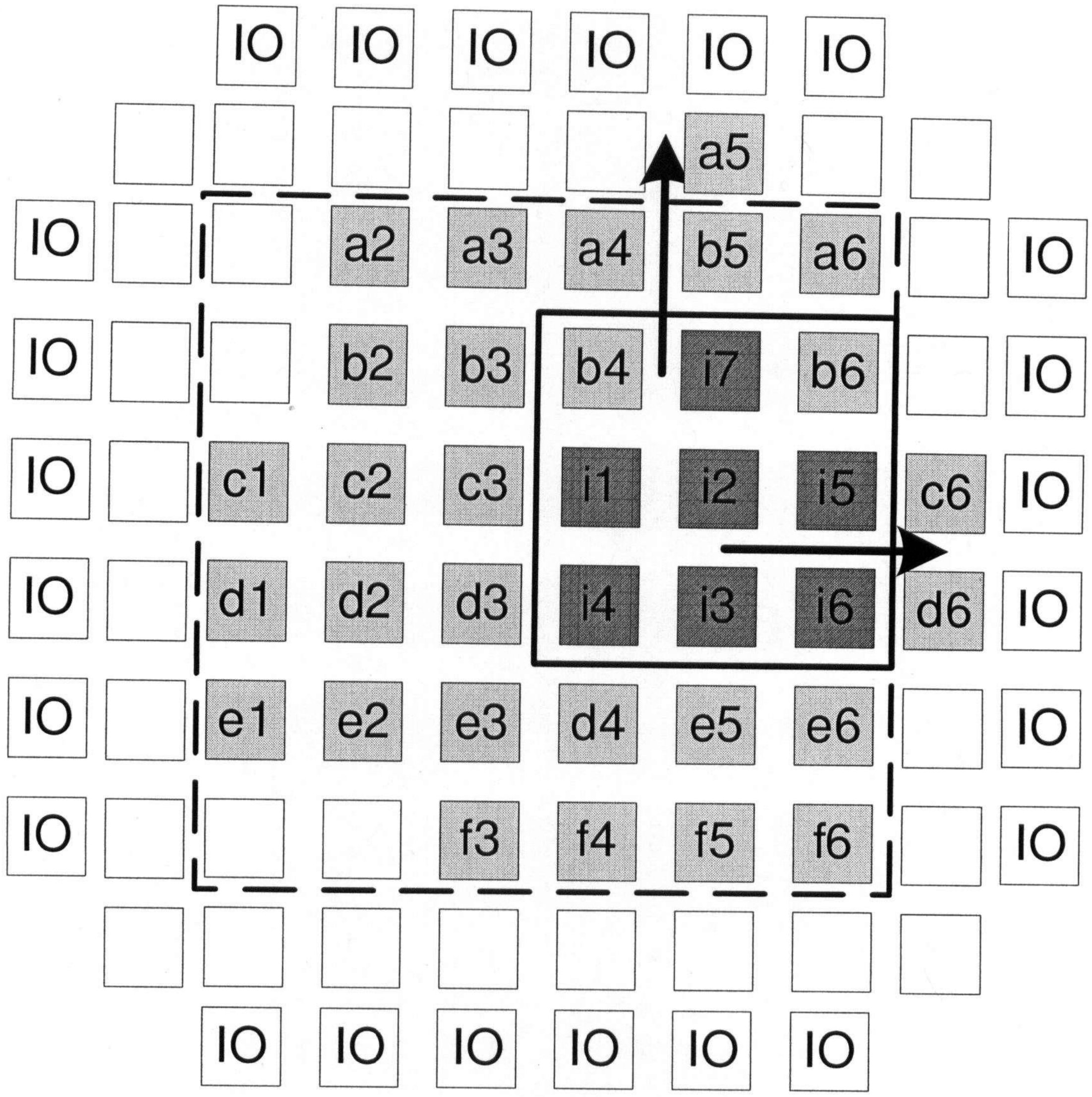

Figure 8 Super Grid Expansion

Referring to the example started in Figure 6, the cells c4, c5, d4 and d5 were marked for removal. These cells will now be replaced with cells i1 to i7. Note that only four free locations are available, but seven new blocks needs to be placed. Blocks i1 to i4 are randomly placed in the free locations without issue. However, there is insufficient room for blocks i5 to i7. To solve the problem of inserting more elements than the amount of free space available in the floorplan region, we use a virtual placement grid called a super-grid that is larger than the physical FPGA size. If the region runs out of space, CLBs to the right of the region are shifted right by 1 CLB location and the floorplan rectangle is increased in width by 1. This is shown in Figure 8, where c6 and d6 are shifted right to make room for i5 and i6. Once the right side is fully shifted by 1 position, the algorithm switches to shifting CLB columns on the top side by 1 position upwards; this is shown in Figure 8 where a5 is shifted up to make room for i7. Whenever needed, the supergrid size array is increased, adding additional rows and columns. Note that the IO locations just shift outwards but are not reordered or increased in number. The super-grid allows the algorithm to shift CLBs to locations *outside* of the normal placement area. This avoids the need for additional calculations to re-shuffle free spaces within a limited placement area, but preserves the *relative* placement of most CLBs with the intent of benefiting from the original spatial locality.

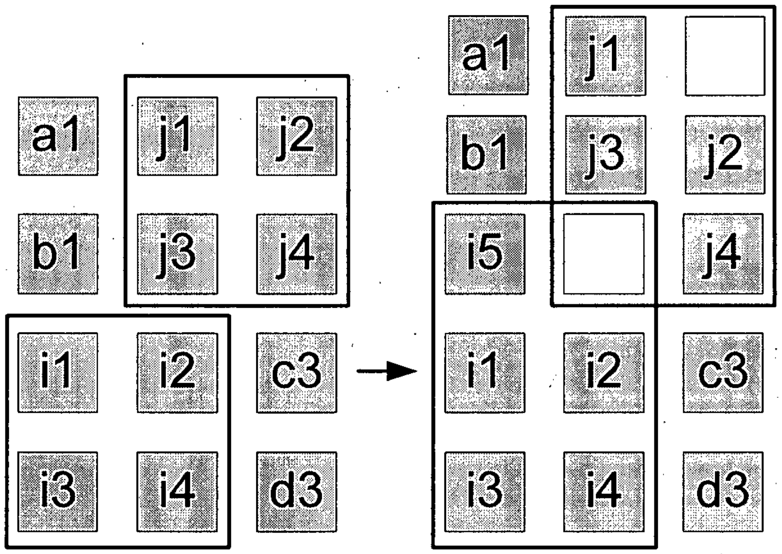

Figure 9 Multi-region floorplan handling

Since iPlace is a multi-region incremental placement algorithm, it must be able to handle multiple floor-planned areas supporting overlaps and expansion of each area individually. To maintain placement locality, the shifting paradigm does not move any affected floor plans when shifting is required. Instead, all affected floor plan regions increase in size along the shifting rows or columns. This idea is graphically illustrated in Figure 9. Two floor planned areas named i and j are shown. If placement region i was to expand upwards by 1 row, floorplan j becomes affected. Instead of moving the entire region j upwards, only the required columns are shifted up (j1 and j3). Both regions i and j increase in size by 1 vertically. Note that regions i and j now overlap. The free element introduced is common to both regions i and j. If region j required more space, it can take advantage of the shifting done for region i and use the newly created free locations.

Observation 1: The shifting paradigm is CPU efficient. It does not need CPU intensive cost function calculations or any sort of iterative location evaluation. The algorithm evenly distributes the expansion across the four sides.

Observation 2: The amount of shifting required to expand a region is quite modest. For example, to expand a 5x5 CLB region by 20%, only one shift on one side is required to make it 5x6. The limited shifting maintains placement locality and does not significantly disturb the overall relative ordering of CLBs in the original placement.

3.4. Compaction

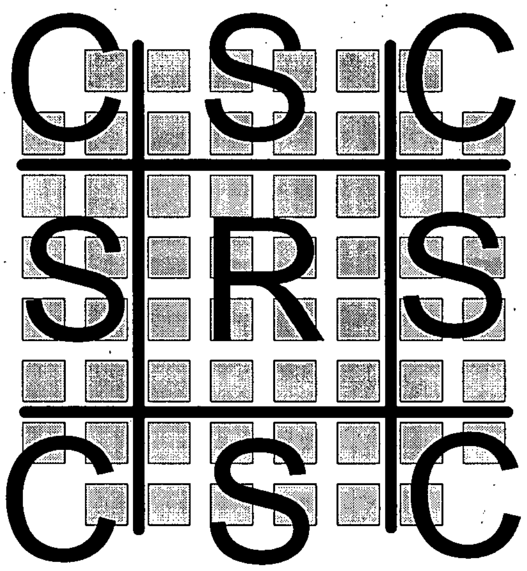

Figure 10 Compaction Regions

The third phase of iPlace is to re-legalize the placement. After the super-grid expansion phase, there could be CLBs located outside of the legal placement area defined by the FPGA array size. One method to re-legalize all CLBs is to use an annealing algorithm. However, this is a slow process and does not guarantee that all CLBs will eventually converge to legal areas. Instead, we propose a *simple* and *fast* solution called "compaction" to overcome this problem. Note that the super-grid can be partitioned into 9 sections like a "#" sign, with the legal placement area at the centre. This is graphically shown in Figure 10 where "R" represents the Regular placement area, S represents Side and C represents Corner. This leaves four corners and four sides to handle. The algorithm works by shifting *all* of the free space (empty CLBs) spread throughout the legal placement region to one end. The algorithm performs horizontal followed by vertical compaction to move free space to the required side (or vice versa for different cases). For the four corners, compaction is done to move the space to the corner. For the four side cases, the free spaces are first percolated to the required side. A centroid position is calculated to estimate where the bulk of the illegal cells are located. The free space is shifted to the centroid location to preserve locality. The centroid location is calculated as the median position for all illegal CLBs located on the side under consideration. Following compaction, the illegal cells are randomly re-inserted into the legal free space. The pseudocode for the compaction algorithm is shown in Figure 11

```
for each illegal region r
   {
   if r is corner
      {
      shift all free space to corner
      randomly move illegal cells in free space
      }
   if r is side
      {
      shift all free space to side
      find centroid of illegal placements in r
      shift free space to centroid
      randomly move illegal cells in free space
      }
   }
```

Figure 11 Compaction Pseudocode

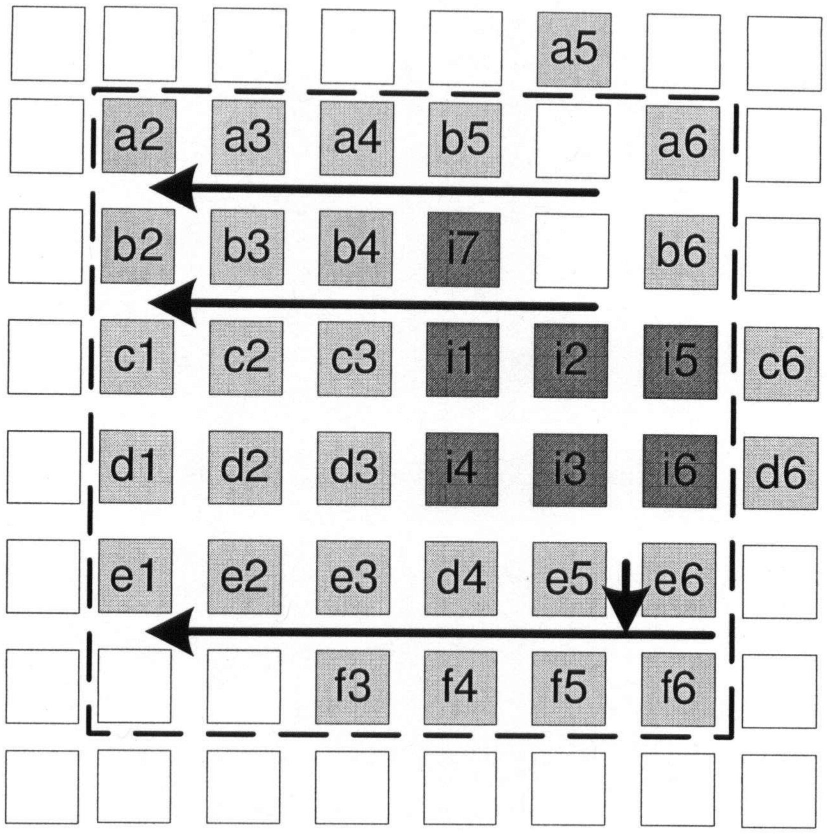

Figure 12 Supergrid Compaction Moves

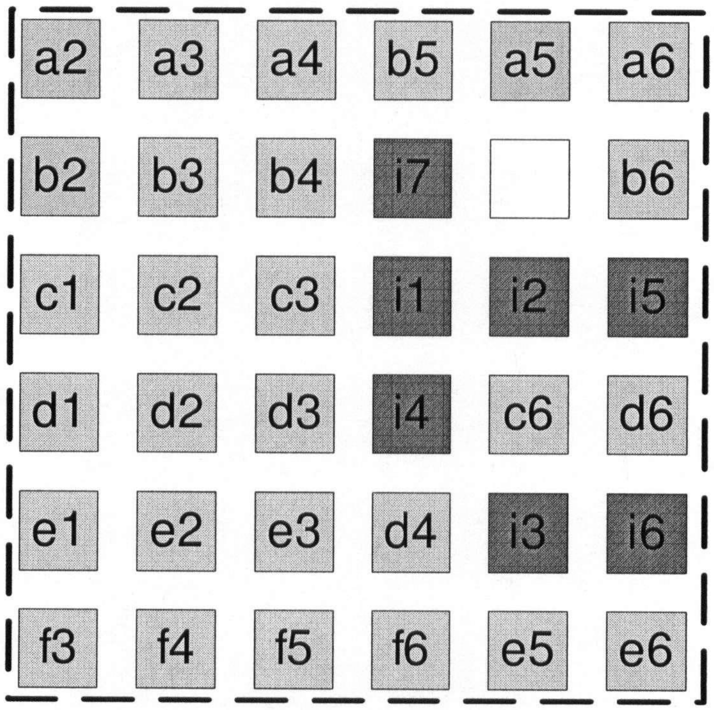

Figure 13 Final Legalized Solution

Continuing with the example from Figure 6 and Figure 8, the compaction process is graphically shown in Figure 12. The block a5 is re-legalized by compacting the free space from the top left corner. Note that the free space has percolated from the top left corner to the locations below a5. The cells c6 and d6 are re-legalized by first compacting the free space from the bottom left hand corner. This percolates the free space to the bottom right hand corner. Next, cells e5 and e6 are compacted downwards to the bottom right hand corner. The final legalized solution is shown in Figure 13.

3.5. Refinement

After the compaction step, we found that the average bounding box and critical path delays were not ideal. In most cases, the bounding box cost reported by VPR was 20% larger than a full placement from scratch. The estimated critical paths were also 10% higher. To refine the solution, we added a low temperature annealing step to iPlace. The refinement phase must not disturb the spatial locality property that iPlace is based upon, but must also be able to perform limited hill climbing to optimize the modified CLBs. To accomplish this task, we re-tuned various parameters within the simulated annealing algorithm of VPR. To limit hill climbing, the *initial temperature* was lowered so that fewer "bad" swaps would be accepted. To maintain spatial locality, the initial *window range* was lowered to focus the swaps within a more localized area. To reduce and control the runtime, the number of swaps per temperature range factor *inner_num* and the temperature degradation factor *alpha* parameters were also tuned.

The initial temperature was selected as the first 44% acceptance rate cross over point during the baseline initial placement. The 44% acceptance rate threshold was chosen based on previous work in [43].

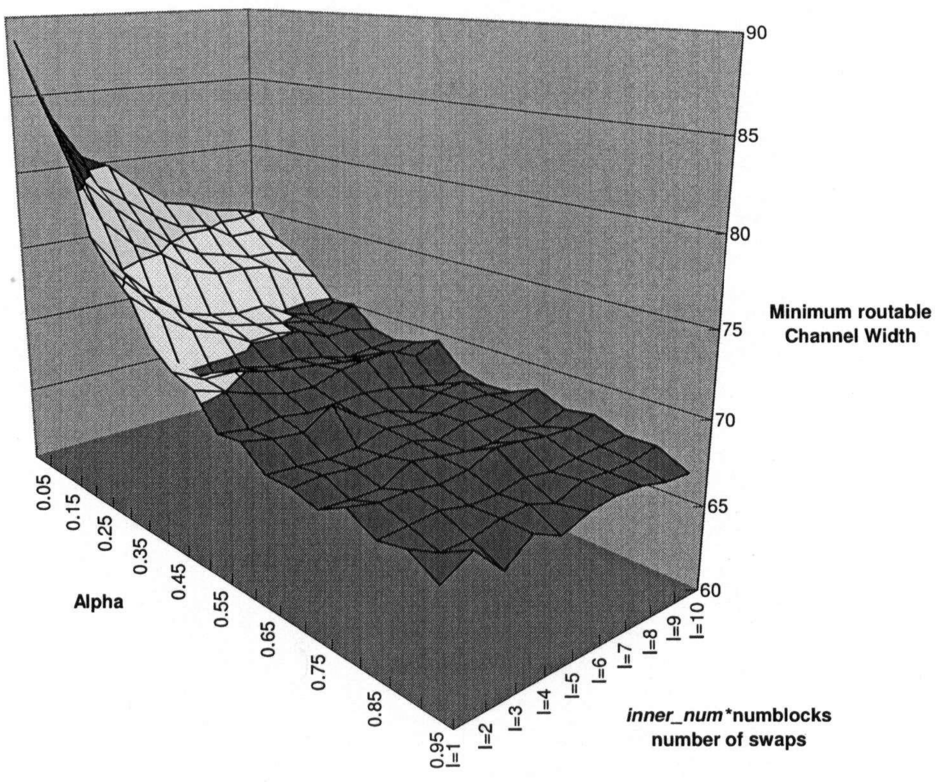

Figure 14 Channel Width tradeoff versus Alpha and Number of Swaps (S)

Figure 14 shows the channel width quality trade off versus temperature degradation (alpha) and the number of swaps *inner_num (I)**numblocks per temperature range. At very low alpha factors, substantially more swaps are required to refine the solution. Very low values represent a rapid cooling schedule. Even with a larger number of swaps, low alpha ranges are unable to produce high quality solutions. There is a 10% channel width degradation when comparing an alpha of 0.95 versus an alpha of 0.05 for a swap multiplier S of 10. A more substantial 50% degradation is observed when an alpha value of 0.05 is used with value S of 1. For the refinement stage of iPlace, a conservative alpha factor of 0.45-0.50 and an *inner_num* multiplier N of 1 to 3 would suffice to produce a high quality channel placement with respect to channel width.

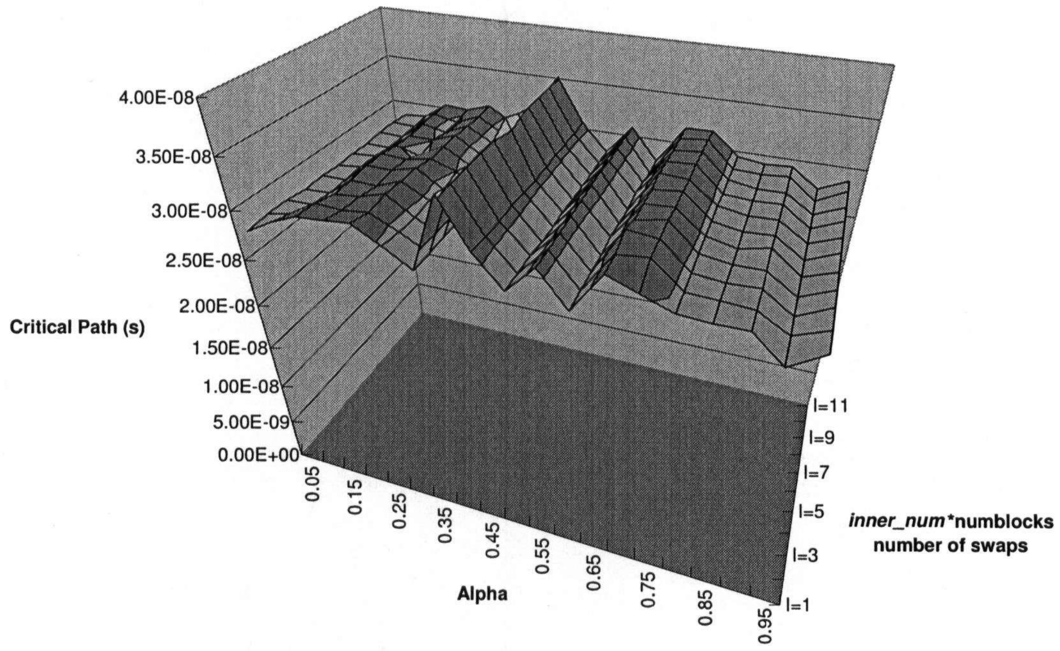

Figure 15 Critical Path tradeoff versus Alpha and Number of Swaps (S)

Figure 15 shows the critical path trade off versus alpha and number of swaps N. The figure shows that the critical path is somewhat noisy for varying values of alpha. The reasoning behind this could be explained by the rapid cooling effect for lower values of alpha. If insufficient hill-climbing is performed, then the solution could be easily trapped by local minimums. This is especially important because iPlace bases the incremental solution on prior placements. If the previous solution was sensitive to begin with, the incremental placement could be greatly affected. From Figure 14, Figure 15 and other tuning trials (not shown), it was determined that an alpha value of 0.7 or above produced the best results.

The final tuned parameters after simulation were found to be the following:

- Initial temperature of 44% acceptance rate from previous placement
- Initial window range of 12.5% of the FPGA width
- Temperature degrading factor *alpha* of 0.7
- Number of swaps per temperature range, *inner_num* of 1 to 3

Based on the tuning, the refinement phase optimizes the placement and produces a high quality result that is comparable to a full placement. The run-time is also very short, and is controllable via the *inner_num* parameter.

3.6. Additional Tuning Considerations

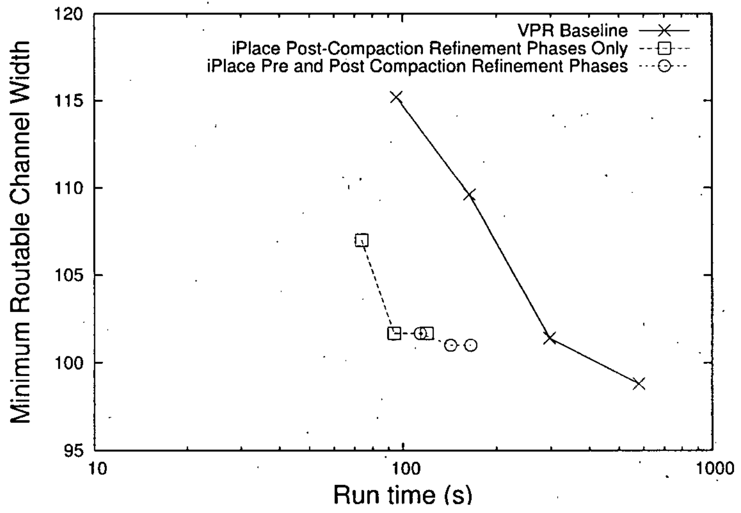

Figure 16 Channel Width versus Run-time Trade off while tuning iPlace

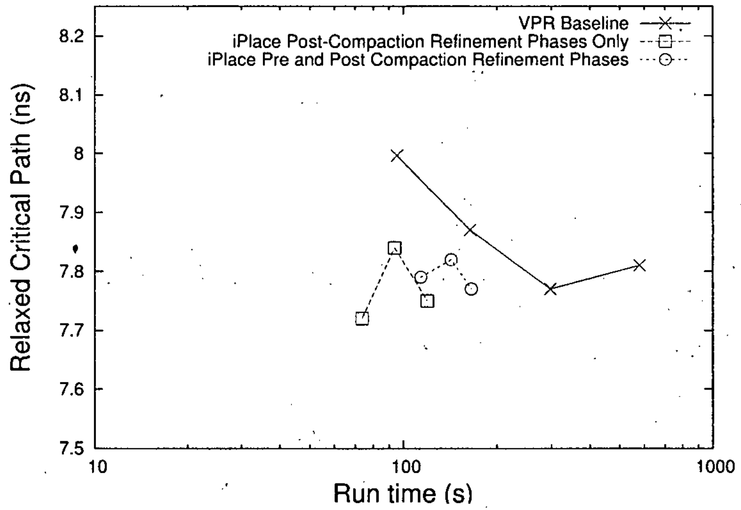

Figure 17 Critical Path versus Run-time Trade off while tuning iPlace

During the design and tuning of the iPlace algorithm, the amount of refinement using simulated annealing was considered a crucial parameter. The refinement stage makes up the bulk of the runtime but is required to produce the highest quality solutions. During the design and tuning of iPlace, it was also considered to have an additional refinement stage in between the expansion and compaction stages. The goal of this additional refinement stage was to reduce the amount of illegal CLBs located outside of the legal placement area. The hope was that with fewer illegal CLBs, the amount of compaction required would be minimized. The pre-compaction refinement stage used the same tuning parameters as the post-compaction refinement stage. The initial temperature was lowered to the 44% threshold, *alpha* was reduced to 0.7 and the *inner_num* parameter varied from 1 to 2 to control the run-time. Tuning was done with 60,000 LUT synthetic circuits produced for [1].

Figure 16 and Figure 17 shows the channel width and critical path versus run-time trade off of having the additional refinement stage before compaction. Figure 16 shows the addition of the pre-compaction refinement stage does help to lower the minimum routable channel width. However, increasing the *inner_num* parameter for the post-compaction refinement stage can make up for the loss of the pre-compaction refinement phase. The crossover point at 100 seconds shows that the post-compaction stage alone can still achieve the same results as having both the pre and post compaction refinement phases but with faster runtime. Figure 17 shows the critical path versus run-time for the same comparison. There are no notable quality differences for the critical path results with the addition of the pre-compaction refinement. Based on these results, it was decided to only have a post-compaction refinement stage.

3.7. Algorithm Conclusions

This chapter has presented an incremental placement algorithm iPlace to be used as part of an incremental CAD flow. The iPlace algorithm was designed based on the principles spatial locality and efficient shifting algorithms. The four steps to the iPlace algorithm include initial placement, controlled expansion, compaction and retuned simulated annealing refinement. The initial placement phase places all unmodified CLBs at their previous placement locations. The expansion phase uses floor-planning and shifting to place all modified CLBs into an expanded placement grid. The compaction phase re-legalizes the placement also by shifting. Finally, the refinement phase produces a high quality incremental placement by cleaning up the intermediate solution with a fast and retuned simulated annealing algorithm.

3.8. Algorithm Limitations

iPlace currently does not take into account that commercial FPGAs have carry chains and hard macro blocks such, memories and multipliers. The current version of VPR is unable to model such constraints. This section will present how we envision handling these cases in the future.

Macroblocks, Memories and Multipliers:

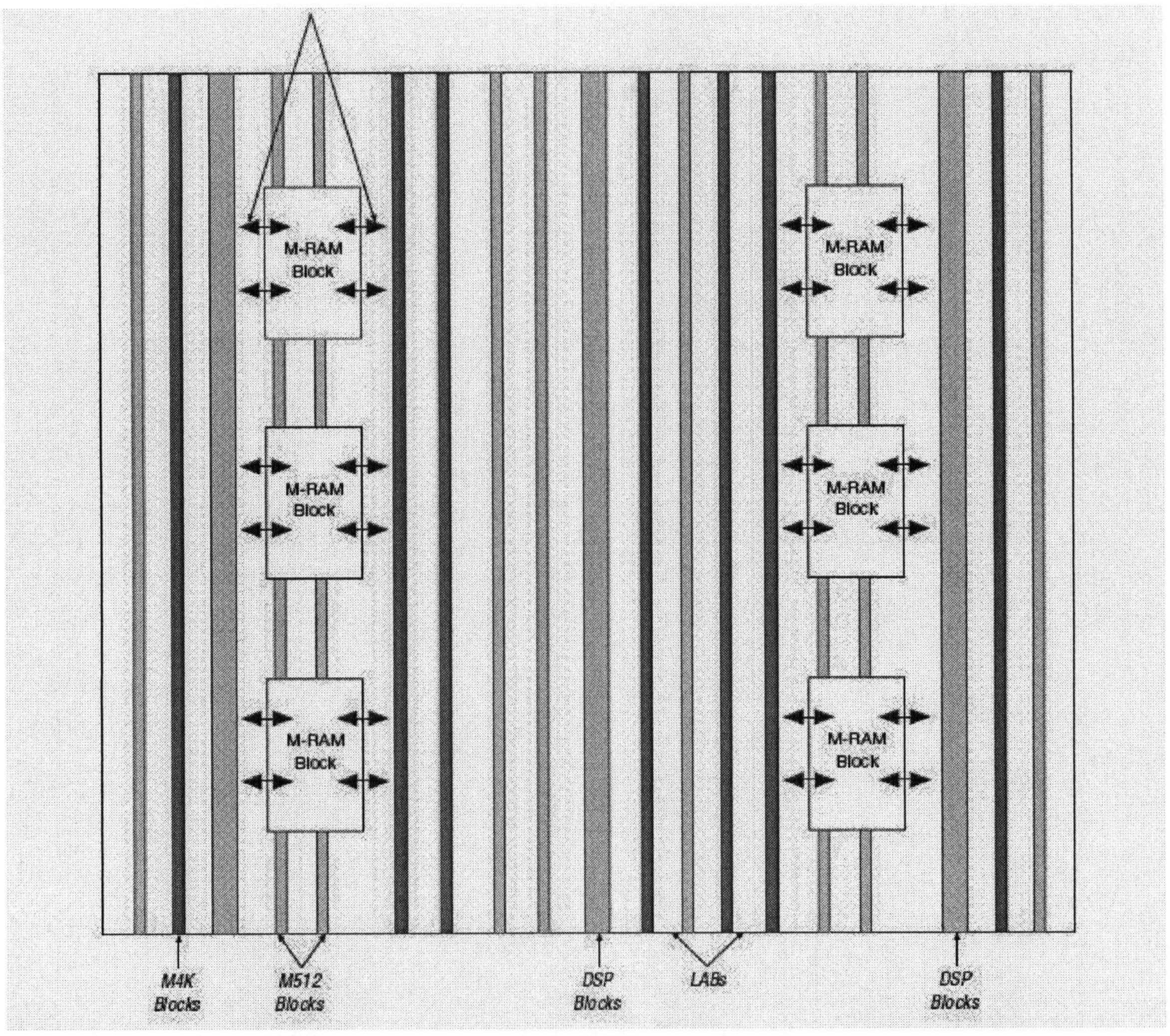

Figure 18 Stratix II FPGA layout [63]

The layout of macro blocks are usually arranged as entire columns in the FPGA array. This is graphically shown in Figure 18 with the layout of an Altera Stratix II FPGA. One way to handle these cases is to divide the placement grid into multiple vertical stripes of CLBs. Each stripe is bounded on the left and/or right by hard macro blocks. By partitioning the FPGA into multiple stripes, each partition can be considered separately for the shifting and super-grid. Other shifting constraints can also be imposed on the shifting algorithm to handle limitations such as the input positions to the macro blocks.

Carry Chains:

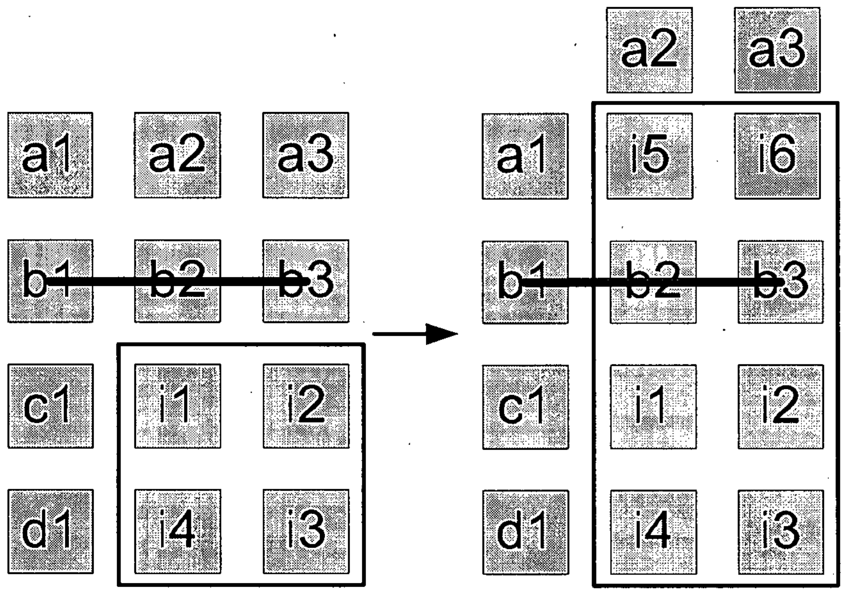

Figure 19 Handling Carry chains

The second constraint currently not handled by iPlace is carry chains. Since carry chains must remain tightly connected, the shifting process cannot destroy these placements. A potential solution to this problem is to lock down the position of carry chains. Taking into account the modest shifting requirements noted in the shifting algorithm, it is possible to fix the carry chain placement location and shift the elements over the carry chain. This is graphically shown in Figure 19. A pre-existing carry chain spans CLBs b1, b2 and b3. An incremental placement region is shown as the bounded box. If the region needs to be expanded to accommodate more elements, the floorplan can be expanded *beyond* the carry chain. The resulting placement does not disturb the carry chain but still allows for the expansion paradigm. The implementation and evaluation of these suggested changes are left for future work.

4. Results

This chapter details the experimental setup and the results obtained when incremental placement is performed using the iPlace algorithm. First, an overview of the experimental goals will be presented. Second, the experimental process and setup will be outlined. Third, the benchmark setup and results for the Single-Region Synthetic benchmark (SYN) set will be presented. Fourth, the benchmark setup and results for the Single-Region Physical Re-Synthesis benchmark (PR) set will be presented. Last, Multi-Region Physical Re-Synthesis benchmark (MR) set will be presented. This chapter concludes with a summary and discussion of the results for incremental placement using iPlace.

4.1. Experimental Goals

Incremental Placement Runtime

The main goal to measure for iPlace is the placement Run-Time (RT) of the algorithm. iPlace is an incremental placement algorithm targeted at reducing the placement run time for iterative development. The main point of comparison will be the runtime required for the baseline VPR toolset to perform a placement solution using default options. In addition, the runtime of the VPR tool set using reduced-quality settings "-fast" *(inner_num=1)* and "-superfast" *(inner_num=0.125)* modes will be contrasted. The "-fast" mode is a standard option available in VPR. The "-superfast" mode was developed in this thesis to compare results against very fast placement.

Minimal Routable Channel Width

The second quality to measure for iPlace will be the Minimal Routable Channel Width (MRCW). The MRCW quality signifies the routability quality for a placement. This is an important factor to consider because commercial FPGA devices have fixed channel widths. If a bad incremental placement solution is created, this could lead to higher channel width requirements that cannot be satisfied by the device. To determine this quality, iPlace will be compared against VPR in default, "-fast" and "-superfast" modes.

Relaxed Critical Path

The third quality to measure for iPlace is the relaxed Critical Path (CP). The relaxed critical path is determined by routing the placement solution at 20% above the minimum routable channel width. This ensures that no portions of the FPGA are severely routing congested to obscure the true critical path. The critical path of a circuit is the longest delay path that a signal has to traverse from an input to an output or between synchronous flip-flops. The critical path determines the clock speed of a circuit, and thus its performance. To determine this quality, iPlace will be compared against VPR in default, "-fast" and "-superfast" modes.

Placement Stability

The last quality to measure is placement stability. Placement stability is a measure of how much the placement has been modified from the "before" circuit state to the "after" circuit state. There is no standard way to measure this, so we have decided to measure the total Euclidean distance traveled in the array ("before" position to "after") by all of the *unmodified* CLBs. The goal of placement stability is to show that incremental placement using iPlace results in a placement closely resembling the initial placement solution.

If an unmodified CLB remains in the same location in the "after" state, the cost for that CLB is 0. For VPR placement from scratch results, we also considered that the array is perfectly symmetrical, so we computed the distance cost total from all possible initial orientations (rotates and flips) of the initial placement and took the lowest total distance. For the iPlace results, we kept only the original orientation. The results presented for placement stability will be *normalized* to a "unit distance" measurement. The unit distance is a summation of the Euclidian distance for all un-modified elements if each element traveled 1 unit horizontally and 1 unit vertically.

The placement stability quality will be presented for the Single-Region Synthetic and Physical Re-synthesis benchmarks. It will not be presented for Multi-Region Physical Re-synthesis benchmarks because significant portions of the circuit are modified.

4.2. Experimental Baseline

The iPlace incremental placement CAD flow is implemented as part of the academic tool VPR [32] and TVPack. TVPack has been modified to include the iRAC [60] clustering algorithm along with the ability to perform incremental re-clustering. The benchmark flow consists of the following parameters and settings:

- Initial benchmark circuit clustering using the iRAC algorithm
- Initial high-quality placement using VPR in default mode
- Single-Region Synthetic benchmark set from Section 4.3
- Single-Region Physical Re-Synthesis benchmark set from Section 4.3
- Multi-Region Physical Re-Synthesis benchmark set from Section 4.4
- FPGA architecture with LUT size ($k = 4$), cluster size ($N = 10$), wire length ($L = 4$), all buffered (bi-dir) routing, TSMC 180nm [64] (PR and SYN benchmarks)

- FPGA architecture with LUT size (k = 6), cluster size (N = 16), wire length (L = 4), all buffered (bi-dir) routing, TSMC 180nm [64] (MR benchmark)
- VPR flags: –verify_binary_search –pres_fac_mult 1.3 –max_ router_iterations 100, relaxed run change: –pres_fac_mult 1.1
- Run-times are for placement only; initialization time is excluded

The benchmarks were incrementally re-clustered by keeping the original clustering solutions for the unmodified CLBs and incrementally re-clustering the modified LEs into modified CLBs. iRAC was used for all clustering because it produces the lowest routed channel widths and delay performance compared to TVPack [25]. The location of the removed CLBs was used to produce a floorplan rectangle as additional input to iPlace.

The CAD flow used to measure the quality of the incremental placer is as follows. The baseline circuits are first clustered using iRAC. The clustered circuits are then placed using the default settings of VPR to obtain a high quality initial placement. Using this initial placement and the floorplan from the benchmark circuit generation process, iPlace is used to incrementally re-place the benchmark circuit. The placement speed of iPlace was varied by setting the *inner_num* annealing parameter to 3, 2.5, 2, 1.5 and 1. Lower values result in faster annealing times, but this does not significantly affect quality.

For comparison, a placement from scratch was also performed using VPR. The *inner_num* parameter was swept with values of 10, 1, 0.5, 0.25 and 0.125. Reducing this parameter reduces the number of swaps that are performed at each temperature. An *inner_num* value of 10 is the "default" value for VPR. An *inner_num* value of 1 is the default when VPR is invoked with the "–fast" placement option. This produces slightly lower-quality placements but increases run-time nearly 10 times. A new "-superfast" option was created by setting the *inner_num* parameter to 0.125. Various other VPR parameters such as initial temperature, range limit etc. were also studied to determine the reduction of run-time versus placement quality trade-off. It was found that reducing *inner_num* provides the most graceful degradation of placement quality versus run time improvement.

The results presented in this chapter consist of the runtime, minimum routable channel width, relaxed critical path and a placement stability analysis. For every placement generated, the VPR binary search routing option was invoked to determine the minimum routable channel width. The relaxed critical path value was determined by routing the placement with 20% more channel width than the minimum required.

All of the simulations were executed on a dedicated Pentium 4, 3 GHz server with 512MB of RAM for each job[10]. Additional memory was not required for the size of the benchmark circuits. Every placement was executed 5 times with 5 different random seeds to reduce the noise in the results. Each datapoint result presented is an arithmetic average of these 5 executions.

[10] Jobs were executed on Westgrid's Glacier cluster and scheduled according to the torque queuing and load balancing system

4.3. Single-Region Synthetic Benchmarks

4.3.1. Benchmark Formulation

The single region synthetic benchmark set is designed to test the performance of the incremental placer with incrementally modified logic. This benchmark set is generated by selecting a subset of a circuit and replacing the subset with a synthetically generated replacement. The synthetic generation and replacement process is discussed in [3] [4].

The process for selecting which elements should be modified is based on the initial placement of the baseline circuit. A random rectangular region is selected and the CLBs from that region are removed and replaced with a synthetic clone circuit. Three different versions of the benchmarks were generated by selecting areas of 2.5%, 5% and 10% of the total CLBs and replacing them with synthetic clones of identical size. For the 2.5% and 5% cases, an additional circuit was generated by doubling the number of CLBs in the replacement clone. In total, this produced 5 "after" circuit states for each original circuit. The 2 doubling cases with 2.5% and 5% more logic were designed to test the iPlace expansion and compaction schemes. The floorplan for this benchmark is the region selected for re-synthesis.

	Original	Synthetic 2.5	
	#CLB	New # CLB	ΔCLB
CLMA	839	839	25
EX1010	107	107	2
MISEX3	140	140	3
PDC	458	458	15
SPLA	369	369	9

Table 2 Synthetic 2.5 Benchmark Characteristics

	Original	Synthetic 5	
	#CLB	New # CLB	ΔCLB
CLMA	839	839	49
EX1010	107	107	9
MISEX3	140	140	9
PDC	458	458	25
SPLA	369	369	22

Table 3 Synthetic 5 Benchmark Characteristics

	Original	Synthetic 10	
	#CLB	New # CLB	ΔCLB
CLMA	839	839	99
EX1010	107	107	15
MISEX3	140	140	15
PDC	458	458	49
SPLA	369	369	49

Table 4 Synthetic 10 Benchmark Characteristics

	Original	Synthetic 2.5d	
	#CLB	New # CLB	ΔCLB
CLMA	839	899	109
EX1010	107	107	5
MISEX3	140	140	9
PDC	458	490	57
SPLA	369	408	64

Table 5 Synthetic 2.5d Benchmark Characteristics

	Original	Synthetic 5d	
	#CLB	New # CLB	ΔCLB
CLMA	839	967	227
EX1010	107	122	26
MISEX3	140	158	33
PDC	458	525	116
SPLA	369	440	120

Table 6 Synthetic 5d Benchmark Characteristics

Table 2 to Table 6 summaries the statistics for a subset of the synthetic benchmark circuits created. The circuits Synthetic 2.5, 5 and 10 are the same sized clones for 2.5%, 5% and 10% rip out areas. The circuits synthetic 2.5d and 5d represent the 2.5% and 5% cut out regions that are replaced with double the number of LEs. The table shows the total number of CLBs for each of the original circuits, the new number of CLBs after the synthetic process and the number of changed (delta) CLBs. This benchmark set demonstrates how iPlace can produce high-quality incremental placements after design changes are made.

The synthetic benchmark process was executed for 20 of the largest MCNC circuits. In this section, the results for a sample of 5 circuits were selected to show the effectiveness of iPlace on a variety of circuit sizes. The full sets of results for the 20 MCNC circuits are provided in Appendix A.

4.3.2. Runtime Results

Synthetic Circuit	Syn - 2.5	Syn - 5	Syn - 10	Syn - 2.5d	Syn - 5d
CLMA	72.0	70.8	73.5	80.3	70.0
EX1010	77.6	75.0	77.0	69.0	76.2
MISEX3	-	-	-	-	-
PDC	80.6	64.0	68.7	84.4	68.1
SPLA	75.7	55.5	44.8	84.0	51.2
				Geometric Mean:	**70.1**

Table 7 Runtime Speedup of iPlace relative to VPR default settings

Synthetic Circuit	Syn - 2.5	Syn - 5	Syn - 10	Syn - 2.5d	Syn - 5d
CLMA	8.3	7.9	8.1	8.9	8.2
EX1010	9.2	8.6	8.8	8.2	8.8
MISEX3	-	-	-	-	-
PDC	9.8	7.0	7.0	10.2	7.9
SPLA	8.7	6.8	5.0	9.7	6.0
				Geometric Mean:	**8.0**

Table 8 Runtime Speedup of iPlace relative to VPR "-fast" settings

Synthetic Circuit	Syn - 2.5	Syn - 5	Syn - 10	Syn - 2.5d	Syn - 5d
CLMA	1.7	1.8	1.7	1.9	2.0
EX1010	2.2	2.0	2.0	2.0	2.2
MISEX3	-	-	-	-	-
PDC	2.2	1.8	2.0	2.6	1.9
SPLA	2.3	1.5	1.2	3.0	1.5
				Geometric Mean:	**1.9**

Table 9 Runtime Speedup of iPlace relative to VPR "-superfast" settings

The runtime speedup achieved with iPlace when compared to VPR in default mode is shown in Table 7 . iPlace is 51 to 84 times faster than VPR in default mode. There is a significant run time improvement when incremental placement via iPlace is used. Table 8 and Table 9 show the speedup comparing iPlace to VPR "-fast" and "-superfast". There is a geometric mean speed up of 8.0 and 1.9 for "-fast" and "-superfast", respectively. It should also be noted that there is no significant slow down for the increased size 2.5d and 5d circuits. The run-time overhead incurred by the expansion and compaction phases are negligible compared to the overall execution time. Entries with a '-' represent run-time results that were *too fast* to be measured reliably (<200ms), so they are omitted from the table.

4.3.3. Channel Width Results

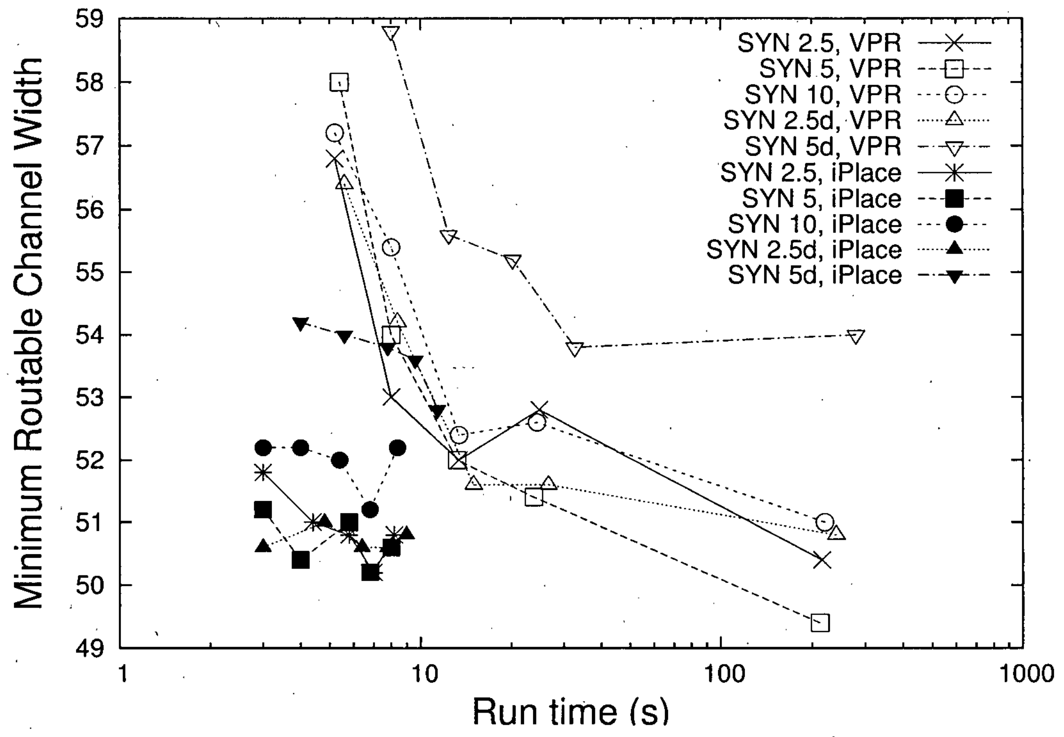

Figure 20: Minimum Routable CW versus Run Time for CLMA, Synthetic

Figure 20 shows the minimum routable channel width versus placement run time for CLMA in the synthetic benchmark set. A channel width degradation of 15-20% is observed with VPR as the run time *(inner_num)* is reduced. In contrast, iPlace produces consistently high-quality solutions. The channel widths for iPlace exceed or are equivalent to default VPR but with 2 orders in magnitude less in runtime. The main conclusion from Figure 20 is that the iPlace curve is *always* below the VPR curve. This means that iPlace is always able to produce better solutions than placement from scratch using VPR. The results were similar for the other benchmark circuits. Full channel width results are provided in Appendix A

4.3.4. Critical Path Results

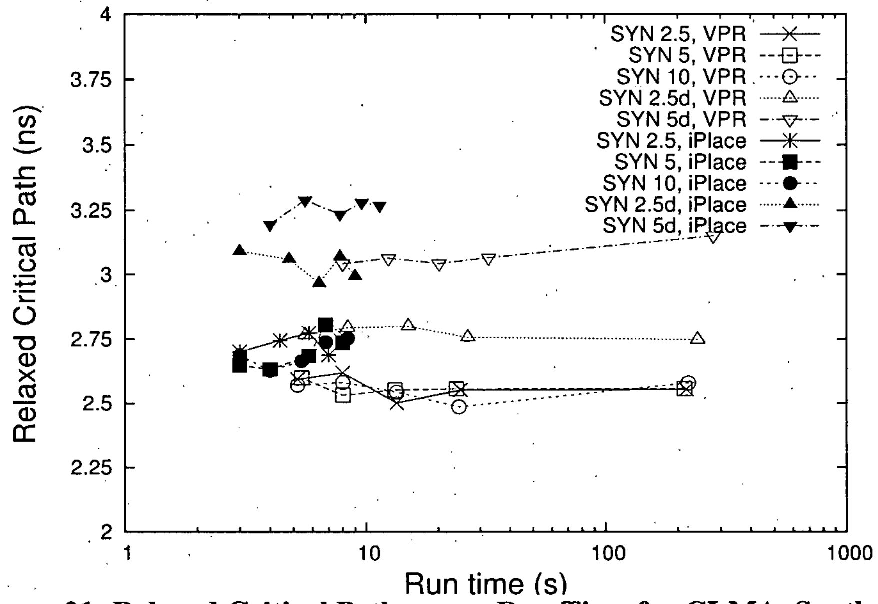

Figure 21: Relaxed Critical Path versus Run Time for CLMA, Synthetic

The relaxed critical path results for the Synthetic CLMA benchmark set is presented in Figure 21. The plot shows that there is a slight critical path degradation (<2%) when comparing iPlace to VPR. When considering the two-orders of magnitude less in run-time for iPlace, it is a small trade off for quality versus run-time. The overall results for all 20 MCNC circuits (Appendix A) show that the critical path results are on par for iPlace relative to VPR. There is less than 1% degradation, which is within error margins. It should also be noted from the previous section that VPR at reduced run-time had significant channel width degradation. Because the relaxed critical path is calculated by routing the circuits at 120% the minimum routable channel width, VPR at lower run-times had even *more* tracks to route with. If the faster VPR placements were routed at the same channel width as iPlace[11], the VPR placement solutions would have to tradeoff routability for higher critical path delays.

[11] Commercial devices have a fixed channel widths

4.3.5. Placement Stability Results

	Syn 2.5	Syn 5	Syn 10	Syn 2.5d	Syn 5d	GeoMean
clma	2.41	2.64	2.05	5.24	3.66	3.02
ex1010	2.36	1.96	2.47	2.84	2.86	2.48
misex3	1.78	2.33	1.72	1.80	2.62	2.02
pdc	4.01	3.09	2.24	4.64	4.41	3.55
spla	3.09	3.77	4.63	3.83	5.27	4.05
					Geomean	2.93

Table 10 Average Displacement Results for Synthetic Benchmark Circuits, Baseline VPR Default

	Syn 2.5	Syn 5	Syn 10	Syn 2.5d	Syn 5d	GeoMean
clma	2.73	2.69	2.72	3.19	3.98	3.02
ex1010	1.53	1.60	1.52	1.88	1.50	1.60
misex3	1.35	1.47	1.47	1.50	2.14	1.57
pdc	2.11	2.11	2.14	2.91	3.05	2.43
spla	1.66	1.69	1.87	2.30	2.91	2.04
					Geomean	2.06

Table 11 Average Displacement Results for Synthetic Benchmark Circuits, iPlace

The placement stability results for the single region synthetic benchmarks are presented in Table 10 and Table 11. The results for VPR and iPlace are normalized to the unit (diagonal) distance measurement. The results show that every unmodified CLBs will travel on average 2 unit distances when incrementally placed with iPlace. In contrast, VPR placement from scratch will travel 3 unit distances on average.

For the circuits Syn 2.5, 5 and 10, the placement stability results for iPlace are similar across the different variations. This suggests that the iPlace placement solution is able to more closely resemble the original placement solution even for different variations in the synthetic flow. In comparison, placement solutions produced with VPR shows larger fluctuation, meaning the placement solutions vary from one annealing run to another. Another result to consider is to consider Syn 2.5 and 5 circuits versus Syn 2.5d and 5d circuits. For iPlace, doubling the synthetic region increases the average displacement per CLB by 0 to ~1.0 units. Placement stability for VPR also increased by approximately the same amount[12], but the results still shows that iPlace produces placements with better stability results.

Overall, the results suggest that iPlace does a good job at preserving placement stability. The placement solution produced by iPlace more closely resembles the previous solution when compared to placement from scratch.

4.3.6. Conclusions for Synthetic Benchmarks

The results presented for Single-Region Synthetic benchmarks show that iPlace is a fast and high quality incremental placement algorithm. The full results for all 20 synthetically modified MCNC circuits are presented in Appendix A. iPlace achieves a speedup of 70 times faster than placement using default VPR settings and 8 times faster than VPR with "–fast" settings. The average of the normalized channel width comparing VPR default placement and iPlace is 1.01. This suggests that iPlace is 1% better than VPR. The placement stability results also show that iPlace is superior to VPR. Unmodified CLBs do not travel as far from their previous placement location when incrementally placed with iPlace. In summary, iPlace is 70 times faster than VPR default placement with no channel width penalty and better placement stability results.

[12] Except for CLMA, the 19 other MCNC circuits had similar results.

4.4. Single-Region Re-synthesis Benchmarks

4.4.1. Benchmark Formulation

	org.	Physical Resynthesis 2.5		Physical Resynthesis 5	
	#CLB	New #CLB	ΔCLB	New #CLB	ΔCLB
CLMA	839	846	32	851	57
EX1010	460	463	12	467	32
MISEX3	140	N/A	N/A	143	12
PDC	458	461	12	465	32
SPLA	369	372	12	376	32

Table 12 PR 2.5 and 5 Benchmark Statistics

	Original	Physical Resynthesis 10		Physical Resynthesis 15	
	#CLB	New #CLB	ΔCLB	New #CLB	ΔCLB
CLMA	839	857	87	876	182
EX1010	460	472	57	478	87
MISEX3	140	N/A	N/A	147	32
PDC	458	470	57	476	87
SPLA	369	381	57	387	87

Table 13 PR 10 and 15 Benchmark Statistics

The Single-Region Physical Resynthesis benchmark set is designed to test iPlace with re-synthesis flows. This benchmark set is generated using the physical resynthesis CAD flow presented in [1]. This flow is an iterative congestion reduction algorithm. It identifies the most congested regions of a circuit and reduces the number of LEs packed per CLB in that region. To generate a set of benchmark circuits, the flow selects the single most congested area and reduces the maximum cluster utilization from 10 to 8 LEs. In effect, this increases the number of CLBs in the changed region by 20%, but the final circuit is still functionally-equivalent to the original. Five benchmark circuits form the original "before" state. Four variations of each circuit were created by selecting a congested region size of 2.5%, 5%, 10% and 15% the total number of CLBs in the circuit. The circuit statistics for the total number of CLBs "after" reclustering, as well as the number of CLBs in the changed region, are shown in Table 12 and Table 13. The floorplan is generated as the congested region selected by the CAD flow. Please note due to discretization effects with the small circuit MISEX3, the 2.5% and 10% were equivalent to the 5% and 15% changes. Therefore, only the latter ones were used. This benchmark set helps demonstrate how iPlace preserves placement quality when used within iterative improvement algorithms.

4.4.2. Runtime Results

Physical Resynthesis Circuit	PR 2.5	PR 5	PR 10	PR 15
CLMA	70.3	64.8	76.9	67.9
EX1010	59.3	74.4	46.8	62.8
MISEX3	N/A	-	N/A	36.0
PDC	63.3	80.2	69.2	70.8
SPLA	257.0	106.5	109.5	44.8
			Geometric Mean	**71.9**

Table 14 Runtime Speedup of iPlace relative to VPR default settings

Physical Resynthesis Circuit	PR 2.5	PR 5	PR 10	PR 15
CLMA	7.6	7.4	8.4	7.8
EX1010	6.7	8.4	5.4	7.2
MISEX3	N/A	-	N/A	4.0
PDC	6.8	8.4	7.8	8.2
SPLA	28.0	12.0	14.5	5.2
			Geometric Mean	**8.1**

Table 15 Runtime Speedup of iPlace relative to VPR "-fast" settings

Physical Resynthesis Circuit	PR 2.5	PR 5	PR 10	PR 15
CLMA	1.7	1.7	1.9	1.5
EX1010	1.7	2.0	1.3	1.7
MISEX3	N/A	-	N/A	-
PDC	1.8	2.0	2.0	2.2
SPLA	6.0	2.5	3.0	1.6
		Geometric Mean		**2.0**

Table 16 Runtime Speedup of iPlace relative to VPR "-superfast" settings

The normalized runtime speedups for the single region physical re-synthesis benchmarks are presented in Table 14 to Table 16. Similar to the synthetic benchmarks from the previous section, iPlace is significantly faster compared to VPR placement. When comparing iPlace to VPR placement using default options, iPlace is 46 to 257 times faster. On average, iPlace is 71.9 times faster than VPR placement from scratch. Compared to the faster placement "-fast" and "-superfast", iPlace is 8.1 and 2 times faster respectively. The lone exception where VPR was faster than iPlace is for the MISEX3 VPR "-superfast" versus iPlace results. iPlace had an average run-time of 0.2 seconds while VPR was too fast to measure. A more accurate timer is needed to gauge the results for small circuits such as MISEX3. Entries with a '-' represent run-time results that were *too fast* to be measured reliably (<200ms), so they are omitted from the table.

4.4.3. Channel Width Results

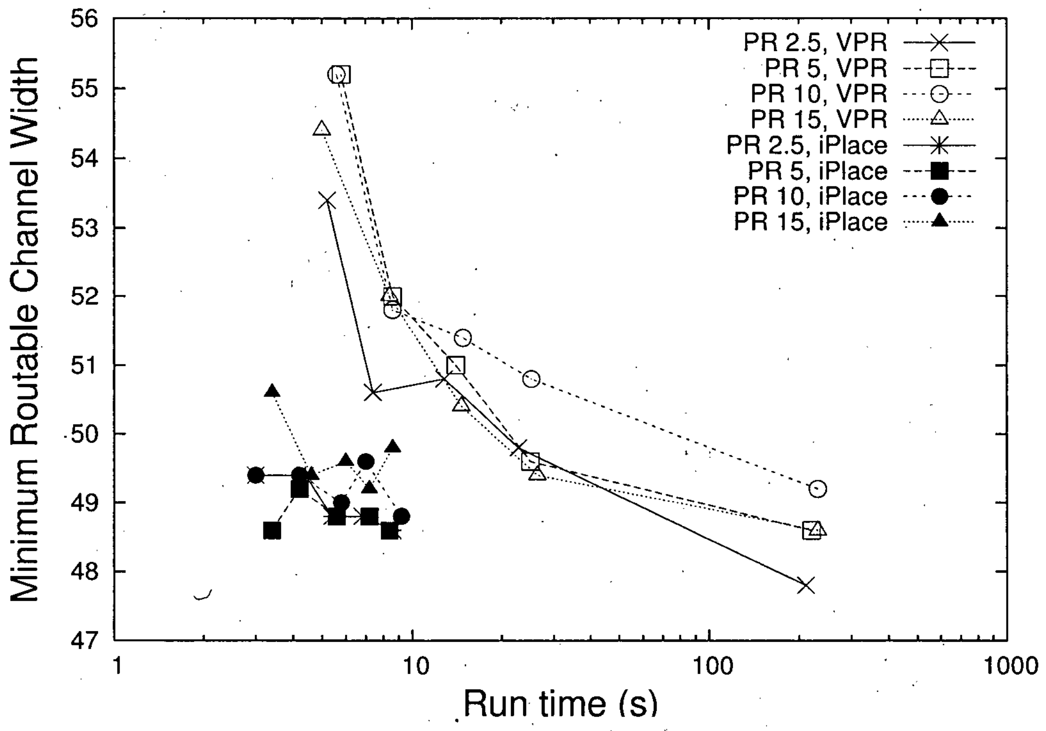

Figure 22 Min. Routable Channel Width vs. Run Time for CLMA, PR

Figure 22 shows the minimum routable channel width versus placement runtime for the CLMA variants in this benchmark set. The reduction of the *inner_num* parameter for VPR placement results in a 10% increase in the minimum routable channel width. The trade-off for 10% channel width increase is a 40-fold decrease in run time. In comparison, iPlace is consistently able to place the circuit with a channel width comparable to "default" VPR placement but with vastly improved runtime (2 orders of magnitude). In fact, iPlace *always* beats VPR in the quality/run-time tradeoff curve. For example, iPlace with an *inner_num*=1 took 3.2 seconds, whereas VPR placement with *inner_num*=0.125 took 5.2 seconds and had significant quality degradation. Full channel width results for this benchmark set are provided in Appendix B.

4.4.4. Critical Path Results

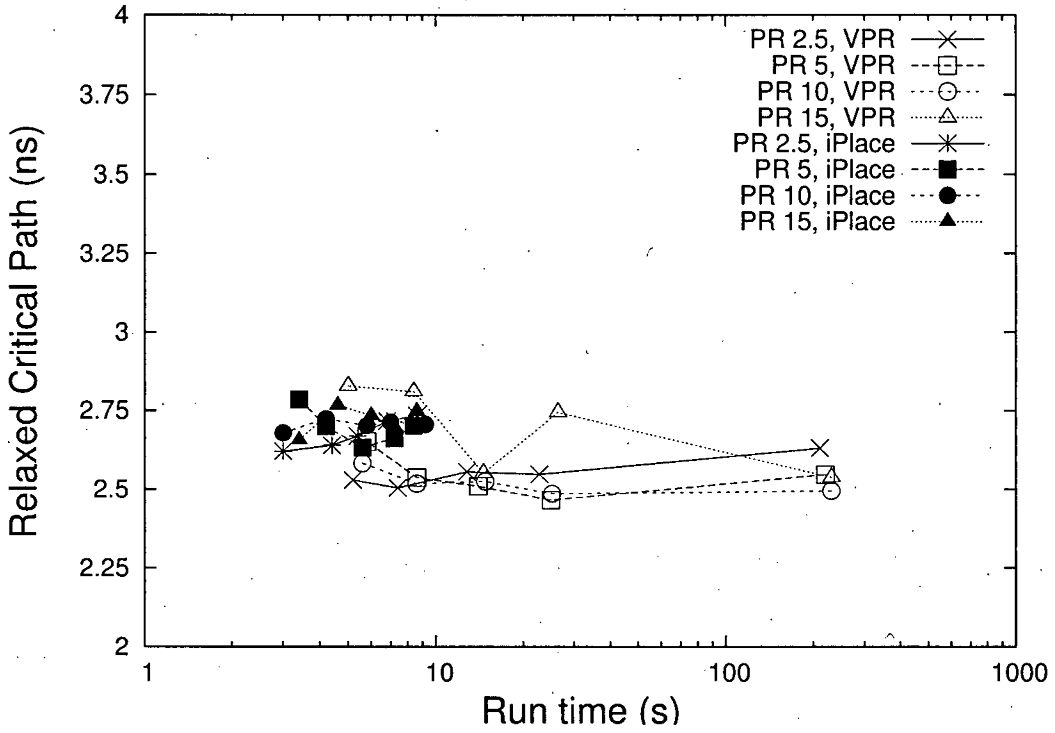

Figure 23 Relaxed Critical Path versus Run Time for CLMA, PR

Figure 23 shows the relaxed critical path results for the CLMA circuits in the PR set. The critical path for iPlace was sometimes worse than VPR by 3-4% for CLMA, but the geometric mean of the critical path over all the circuits was on par. Figure 23 shows that the critical path does not degrade with the reduction in run-time for both VPR and iPlace. However, this may not hold true with real (fixed channel width) devices. For the VPR results, the channel width increased by 20%, which means the relaxed critical path has more tracks available to route, which may help the router optimize the critical path a bit more. In comparison, iPlace uses a similar number of routing tracks for all cases, yet is still able to preserve critical path delay.

4.4.5. Placement Stability Results

	PR 2.5	PR 5	PR 10	PR15	Geomean
CLMA	2.94	3.38	5.30	2.82	3.49
EX1010	2.74	2.56	2.54	3.25	2.76
MISEX3	N/A	2.26	N/A	2.58	2.42
PDC	5.14	3.41	4.43	4.48	4.32
SPLA	3.86	3.87	4.18	3.44	3.83
				Geomean	3.29

Table 17 Average Displacement Results from PR Benchmark Circuits, VPR "Default"

	PR 2.5	PR 5	PR 10	PR15	Geomean
CLMA	2.74	2.76	4.46	2.94	3.15
EX1010	1.70	1.78	1.79	1.79	1.77
MISEX3	N/A	1.55	N/A	2.07	1.79
PDC	2.34	2.45	2.62	2.67	2.52
SPLA	1.76	1.93	1.88	2.25	1.95
				Geomean	2.18

Table 18 Average Displacement Results from PR Benchmark Circuits, iPlace

The placement stability results for the Physical Re-Synthesis benchmark set are presented in Table 17 and Table 18. The results show on average, iPlace travels 2.2 unit distance per unmodified CLB compared to a distance of 3.3 for VPR placement from scratch. This shows that iPlace does preserve the previous placement solution better than a VPR placement from scratch. However, the results for the re-synthesis benchmark set were not as good as the synthetic set. This is due to the nature of the re-synthesis process. The congestion reduction flow always creates 20% more CLBs due to the white space insertion. To make room for the newly created CLBs, iPlace must use the expansion and compaction schemes to fit all of the CLBs into the floor-planned region. This causes the extra shifting of the unmodified CLBs further away from their starting positions.

4.4.6. Conclusions for Single-Region Re-synthesis Benchmarks

The results presented for the single-region re-synthesis benchmark set also shows that iPlace is a fast incremental placement algorithm that produces high quality incremental placements. Overall, iPlace is ~72 times faster than default VPR in default mode and about ~8 times faster than "–fast" mode while achieving with similar quality channel width (CW) and critical path (CP) results. To push VPR even further, the "–superfast" mode (*inner_num*=0.125) was also added for comparing iPlace and VPR. At this point, iPlace is still twice as fast and does not show the quality degradation exhibited by VPR. Notice that some run-time results were *too fast* to be measured reliably (<200ms), so they are omitted from the table (shown as a dash). The full set of run-time, channel width and critical path, quality results are provided in Appendix B.

4.5. Multi-Region Re-synthesis Benchmarks

4.5.1. Benchmark Formulation

	org. #CLB	Multi Region - 50			Multi Region - 40		
		New #CLB	ΔCLB	Num. Regions	New #CLB	ΔCLB	Num. Regions
CLONE	3151	3618	2233	135	3310	762	46
STDEV0	3148	3603	2218	114	3595	2208	114
STDEV010	3152	3463	1490	85	3278	588	37

Table 19 Multi Region 50 and 40 Benchmark Characteristics

	org. #CLB	Multi Region - 30			Multi Region - 20		
		New #CLB	ΔCLB	Num. Regions	New #CLB	ΔCLB	Num. Regions
CLONE	3151	3265	560	29	3206	275	12
STDEV0	3148	3606	2224	116	3272	617	30
STDEV010	3152	3254	490	29	3193	202	9

Table 20 Multi Region 30 and 20 Benchmark Characteristics

	org. #CLB	Multi Region - 10		
		New #CLB	ΔCLB	Num. Regions
CLONE	3151	3288	681	34
STDEV0	3148	3370	1087	50
STDEV010	3152	3237	425	20

Table 21 Multi Region 10 Benchmark Characteristics

The Multi-Region Physical Re-synthesis (MR) set of benchmarks is designed to test the performance of the incremental placer with multiple incrementally modified regions. Multi-region incremental placement allows designers to make changes in multiple parts of a circuit and still be able to incrementally re-compile the design in a quick and efficient manner. This set of benchmarks is also generated using the physical re-synthesis flow outlined in [1]. Instead of identifying the most congested area, the flow also supports identifying multiple congested regions. The circuits used for this experiment are synthetically generated with varying levels of congestion. The size of each circuit is ~50,000 logic elements [1]. To create a family of benchmarks, 5 variations of 3 circuits were created. The five variations are produced by targeting a percentage reduction of 10%, 20%, 30%, 40% and 50% in the minimal routable channel. This is accomplished by using a re-synthesis algorithm to re-cluster parts of the circuit. The algorithm begins by identifying the most congested CLB and marking a region within a Euclidian distance of 5. The CLBs within the region becomes a congestion region. The algorithm then identifies the next most congested unmarked region, it iterates to find all congested regions with routing requirements higher than the targeted reduction. Each region can contain a maximum of 25 CLBs, but there may be fewer because regions can overlap. Each CLB can only belong to one region. To achieve the reduction in channel width required, 3 white space LEs are inserted into each cluster identified as congested, reducing the utilization from 16 to 13 LEs. This produces ~20% increase in CLBs per region. The increase in the number of CLBs means that iPlace must use the expansion paradigm to fit the increased number of CLBs into regions that are too small. The floorplan for the multi-region benchmarks is generated based on the identified congested regions. Note that due to overlapping congested regions, some floorplan regions will overlap. The circuit statistics for the MR set of benchmarks are shown in Table 19, Table 20 and Table 21. For each benchmark, the total number of CLBs, the number of "depopulated" delta CLBs and the number of changed regions are shown. This benchmark set shows that iPlace is capable of handling multiple overlapping regions of various sizes.

4.5.2. Runtime Results

Multi-Region Circuit	MR - 50	MR - 40	MR - 30	MR - 20	MR -10
CLONE	62.8	70.0	68.3	61.5	75.9
STDEV0	67.3	66.8	55.1	56.0	66.3
STDEV010	47.1	59.7	68.6	66.5	58.5
				Geometric Mean	**63.0**

Table 22 Runtime Speedup of iPlace relative to VPR default settings

Multi-Region Circuit	MR - 50	MR - 40	MR - 30	MR - 20	MR -10
CLONE	6.7	7.9	7.5	6.6	8.2
STDEV0	7.6	6.9	5.6	6.5	7.1
STDEV010	4.9	6.3	8.0	7.1	5.8
				Geometric Mean	**6.8**

Table 23 Runtime Speedup of iPlace relative to VPR "-fast" settings

Multi-Region Circuit	MR - 50	MR - 40	MR - 30	MR - 20	MR -10
CLONE	1.2	1.5	1.4	1.5	1.3
STDEV0	1.5	1.4	1.1	1.1	1.4
STDEV010	0.9	1.3	1.7	1.4	1.2
				Geometric Mean	**1.3**

Table 24 Runtime Speedup of iPlace relative to VPR "-superfast" settings

The run time results for Multi-Region incremental placement are presented in Table 22, Table 23 and Table 24. These results present the speedup achieved for VPR runtime in default, "-fast" and "-superfast" modes versus iPlace run time. Compared against VPR in default mode, iPlace is 63 times faster. Compared against "-fast" and "-superfast" modes, iPlace is 6.8 and 1.3 times faster respectively. For large changes such as the circuits from the Multi-Region – 50 benchmark set *(Table 19),* up to 2/3 of the circuit is physically re-synthesized and requires re-placement. The results presented above show that the speedup achieved by iPlace does not degrade significantly even for multiple regions spanning 1/3 to 2/3 of the CLBs. The conclusion drawn from the speedup results is that iPlace is a scalable algorithm capable of handling significant changes to a netlist even for large 50,000 LUT circuits.

4.5.3. Channel Width Results

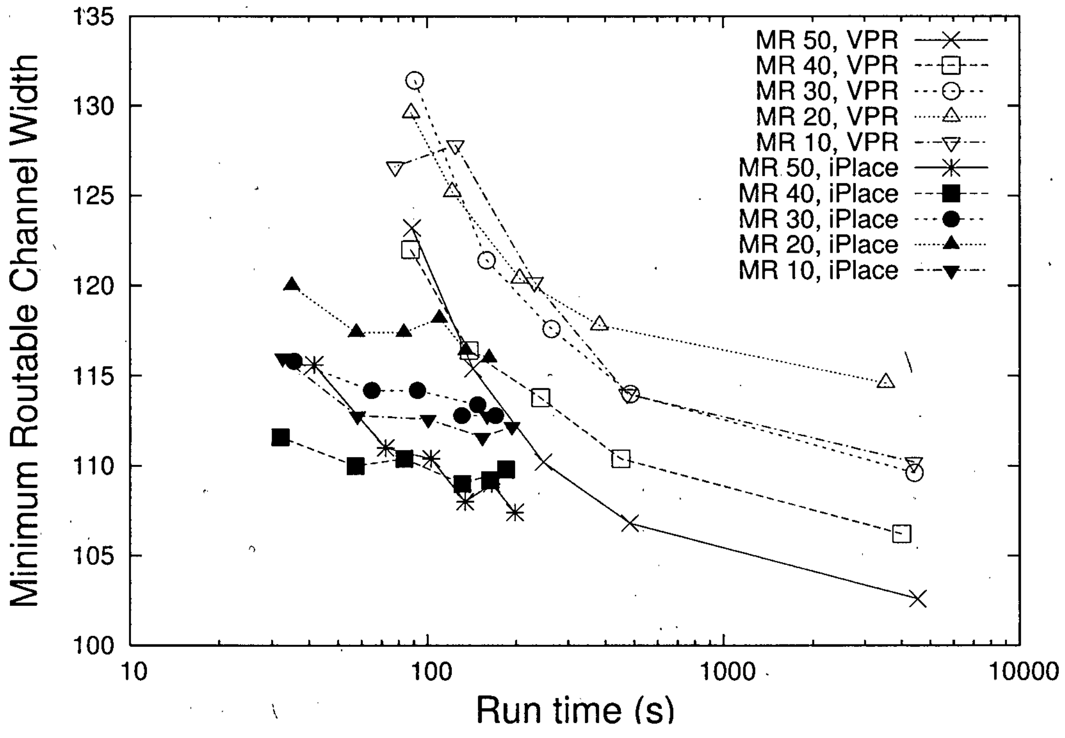

Figure 24 Minimum Routable CW versus Run Time for Stdev010, MR

Figure 24 shows the minimum routable channel width versus runtime for the Stdev010 variants of the MR benchmark suite. Similar results were reached when compared to the single region simulations. As run time is reduced for VPR placement from scratch, the channel width quality degrades significantly. For a 40-50x speed increase, the channel width quality degrades by ~15%. In comparison, iPlace is two orders of magnitude faster in run time compared to a full placement. The quality of the multi region incremental placement is slightly degraded compared to a full placement. There is a 2-4% loss in channel width quality in exchange for the speed up. Compared to a "–fast" placement, iPlace produces similar quality results but is 6-8 times faster. Results for the other MR benchmark circuits were similar. iPlace consistently produces excellent channel width results with significantly reduced run-time.

4.5.4. Critical Path Results

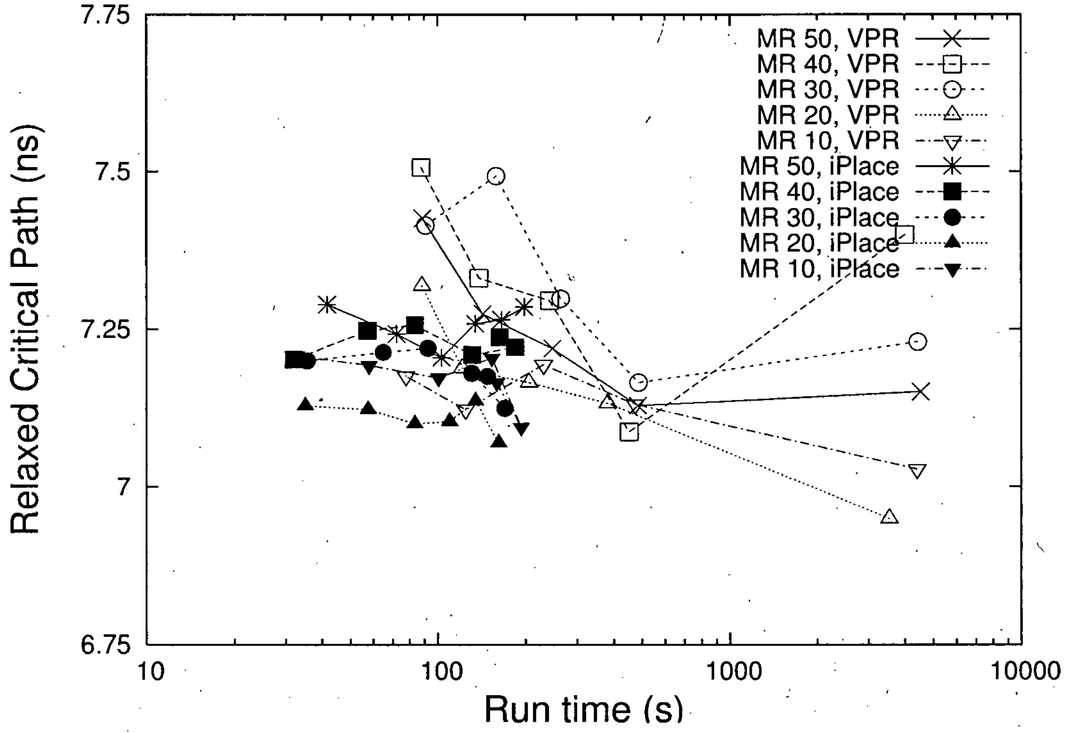

Figure 25 Relaxed Critical Path versus Runtime for Stdev010, MR

Figure 25 shows the relaxed critical path delay results for the Stdev010 circuit in the MR benchmark suite. Similar to the previous results, iPlace produces very competitive results at a fraction of the time. The iPlace critical path results were on par when compared against full and fast placement from scratch using VPR. Results for the other MR circuits were similar. There is no significant critical path increase even for large circuits incrementally placed with multiple modified regions.

4.5.5. Conclusions for Multi-Region Re-synthesis Benchmarks

The results for the Multi-Region Physical Re-synthesis benchmark set were presented in this section. Findings for the MR benchmark set include a 63 times speedup in placement run time when comparing iPlace to VPR in default mode with 2-4% loss in channel width and critical path quality. When comparing the faster VPR "-fast" and "-superfast" modes, iPlace is still 6.8 and 1.3 times faster respectively with on par or better quality for channel width and critical path. A full summary of the multi-region results are presented in Appendix C.

A key finding is that the quality of multi-region incremental placement does not degrade even when a substantial percentage of the circuit is modified. From Table 19, the Multi Region-50 set of circuits have 1/3 to 2/3 of the CLBs modified. However, iPlace is still able to produce quality results due to floor-planning and controlled expansion.

4.6. Experimental Conclusions

This chapter has presented the benchmarking process and results obtained while evaluating iPlace. The aspects used to measure the quality of the incremental placement include the runtime speedup, the minimum routable channel width, the relaxed critical path and the placement stability.

Three benchmarking sets were produced to evaluate iPlace. The first two benchmark sets are used to evaluate the effectiveness of iPlace for typical incremental placement use cases. The third benchmark set is used to evaluate the scalability of iPlace. First, the Synthetic benchmark set simulates design changes by replacing a section of the circuit with a synthetic clone. Second, the Physical Re-synthesis benchmark re-synthesizes part of the clustering solution in order to target a hard FPGA constraint. Third, the Multi-Region Physical re-synthesis benchmark set scales the re-synthesis algorithm to multiple regions of modification and to much larger benchmark circuits.

Speedup	VPR "default" / iPlace	VPR "-fast" / iPlace	VPR "-superfast" / iPlace
Single Region - Synthetic	70.1	8	1.9
Single Region - Physical Resynthesis	71.9	8.1	2.0
Multi-region - Physical Resynthesis	63.0	6.8	1.3
Geometric Mean	68.2	7.6	1.7

Table 25 Multi Region Run-time Speedup Summary

A summary of the run-time speedup obtained with iPlace is presented in Table 25. Overall, the results show that iPlace is significantly faster than placement using VPR. When comparing iPlace to VPR in default mode, iPlace is 63 to 72 times faster in run time. There was no obvious channel width or critical path quality degradation for the single region benchmarks. For the multi-region benchmark set, a 2-4% quality degradation was observed for channel width and critical path. In addition to comparing iPlace to a full placement via VPR, comparisons were also made to the "-fast" *(inner_num=1)* and "-superfast"*(inner_num=0.125)* modes of operation. When comparing iPlace to VPR "-fast", iPlace was 6.8 to 8 times faster. When comparing iPlace to VPR "-superfast", iPlace was 1.3 to 2.0 times faster. In terms of quality, iPlace always produced on par results compared "-fast" and was significantly better than "-superfast". It is also shown that iPlace is a stable placement algorithm. On average, unmodified CLBs travel ~2 CLB units from their previous placement location.

Observation 3:

When examining the channel width to run-time trade off curves (Figure 20, Figure 22, Figure 24), the iPlace curve is always positioned *below* the VPR curve. This indicates that the quality versus run-time trade-off for iPlace is better than VPR. In fact, VPR experiences up to 15-20% degradation in channel width quality as the run-time is reduced by 100 times. In comparison, iPlace is able of producing VPR full placement quality results with the two-orders less in magnitude in run time.

Observation 4:

There was no obvious routing quality degradation with iPlace for the range of *inner_num* values used (1 to 3), but it was observed that values below 1 do provide lower quality results. In fact, lowering *inner_num* too low is equivalent to omitting the annealing step altogether. We observed roughly 10% critical path delay increase and 20% bounding box cost increase when the low temperature annealing step is left out from iPlace. Without annealing, the run time is too fast to measure.

Full Results

The full results obtained for the three benchmarking schemes are provided in Appendix A, Appendix B and Appendix C. The tables show the run time (RT), channel width (CW), critical path (CP), total post-placement bounding box cost (Bbox) and the total post-routing wirelength (WL) of iPlace. Normalized comparisons of VPR default, VPR "-fast", VPR "-superfast" to iPlace are also provided. The columns ending in Q, such as CWQ, show channel width (CW) quality (Q) normalized to iPlace.

5. Qualitative comparisons

We believe that there are *no* other general incremental placement algorithms available for academic FPGA research, so we are unable to present head-to-head comparisons in terms of runtime and quality results. Instead, this section will present a "qualitative" comparison of relevant incremental placement algorithms and fast placement approaches.

The most relevant incremental placement algorithm is the Singh and Brown ICP placer [54][13]. The authors benchmarked the ICP placer with ~15% modified LEs to several MCNC circuits. The paper reported an 8x speedup compared to VPR. Also, they analyzed the run-time complexity of the algorithm and reported that as the number of modified LE increases, the algorithm will eventually be slower than a full placement by VPR. In comparison, this thesis has shown that iPlace is a scalable algorithm capable of handling *large* multi-region modifications. iPlace achieves a 63 times speed up even when up to 2/3 of a circuit is modified and incrementally placed. The key differentiation between iPlace and ICP is that iPlace uses floor-planning and partial design shifting. These simple algorithms are CPU efficient compared to proposing and evaluating cost changes for every LE or CLB swap.

Other incremental placement algorithms presented in Section 2.3 did not present speedup results. We believe that iPlace is the first fully general purpose incremental placement algorithm available for FPGAs. We hope that iPlace will be followed by future incremental placement research that can be compared to the results found by this thesis.

[13] We contacted the authors, but were unable to obtain benchmarks or source code for comparison

Because there are no other suitable *incremental* FPGA placement results to compare, we compare iPlace to other *fast* full-placement schemes. The ultra-fast placement algorithm by Sankar achieves 52x speedup and increases bounding box cost 33% over default VPR [61][14]. Mulpuri compared critical path versus runtime of different placement algorithms, showing ~10x speedup reduces quality by ~30% [60]. The Frontier system by Tessier computes a good floorplan of pre-designed macro blocks followed by a low-temperature anneal, improving both placement run-time by ~17x and critical path by ~10% versus Xilinx software [62]. In contrast, this thesis achieves 35-260 times speedup with no increase to critical path, channel width, or bounding box.

Overall, we believe that iPlace is a unique algorithm for incremental FPGA CAD flows. In comparison to other FPGA incremental algorithms, iPlace is more scalable and faster when compared to VPR. The results found in this thesis showed an approximate 60x speedup for iPlace versus an 8x speedup for the ICP algorithm. Although academic research in this area is limited, we hope future work will provide additional comparisons. Compared to fast placement algorithms, iPlace shows similar speedups but do not exhibit the quality degradations suffered as a trade-off for speed. iPlace is able to generate the same high quality placement as a full placement from scratch.

[14] Sankar used an older version of VPR that is not timing-driven.

6. Conclusions and Future Work

In this thesis, we have presented a new incremental placement algorithm iPlace that significantly reduces the placement time for changes to an already placed circuit. The key ideas contributing to this algorithm include the use of an *initial placement, floor planning, shifting* and a *placement super-grid.* The iPlace algorithm consists of four steps. The first step is the use of an initial placement and floor-planning. The second step is the insertion of modified CLBs using the placement super-grid. The third step is the re-legalization of the placement through compaction. The final step is a short simulated annealing refinement to optimize the solution.

Three suites of benchmarks circuits were designed to determine the performance of iPlace. First, the single-region synthetic set simulates design changes by significantly modifying a region of logic using the Perturber + Mutator flow [3][4]. Second, the single-region physical re-synthesis set simulates a re-synthesis change to target a channel width constraint. Using the Un/DoPack flow [1] where white space is inserted into fully packed CLBs. Third, the multi-region physical re-synthesis benchmark set simulates multiple design changes across multiple regions for large benchmark circuits. Multiple "congested" regions were selected and depopulated using Un/DoPack to target a percentage reduction in the minimum channel width.

Speedup	VPR "default" / iPlace	VPR "-fast" / iPlace	VPR "-superfast" / iPlace
Single Region - Synthetic	70.1	8	1.9
Single Region - Physical Resynthesis	71.9	8.1	2.0
Multi-region - Physical Resynthesis	63.0	6.8	1.3
Geo.Mean	68.2	7.6	1.7

Table 26 Overall Run-time Speedup Summary

A summary of the results achieved using iPlace is presented in Table 26. Table 26 presents the speedup achieved using iPlace compared to VPR in "default" mode, VPR in quality reduced "-fast" mode and VPR in a newly created "-superfast" modes.

For single region synthetic benchmarks, it was found that iPlace is 70 times faster than VPR in default mode with *no channel width or critical path* degradations. Even when compared to the quality reduced "-fast" and "-superfast" modes of VPR, iPlace is still 8.0 and 1.9 times faster respectively.

For single region physical re-synthesis benchmarks, it was found that iPlace is 71.9 times faster than VPR in default mode. The results and speedups observed for the single-region physical re-synthesis benchmark set were similar to the single-region synthetic benchmark sets.

When considering multi-region incremental placement, iPlace is a very competitive algorithm. With 1/3 to 2/3 of a circuit modified, iPlace produces results with less than 4% loss in quality but 63 times faster compared to a full VPR placement. Compared to VPR "-fast" and "-superfast" placement, iPlace is 6.8 and 1.3 times faster with no loss in quality. This shows that iPlace is an algorithm capable of scaling to significant modifications throughout a circuit.

On average, iPlace is 68.2 times faster than VPR in default mode with negligible degradation in placement quality. This is a *significant* performance increase considering that VPR in "-superfast" mode, which is still ~2x slower, results in a significant channel width degradation of ~15-20%.

Finally, we believe that iPlace is the *first* incremental placement algorithm for FPGAs capable of handling multiple regions of incremental modification with substantial run-time improvements and no quality degradations.

6.1. Future work

This thesis concludes with a summary of future follow up work to the iPlace algorithm. The iPlace algorithm is currently implemented as part of VPR and lacks support for several elements found in commercial FPGAs. Also, the algorithm is simplistic, and may be improved upon with further extensions to the shifting, compaction and refinement phases. Finally to provide an end to end incremental flow, incremental routing should also be considered.

6.1.1. Support for Macro Blocks

Commercial FPGAs contain macro blocks such as DSPs, memories and multipliers. The current academic toolset available (VPR) has not been extended to model such elements. Future research for FPGAs will hopefully extend the academic framework to support such elements. Once the underlying structures have been modeled, iPlace can then be modified to implement the proposed solutions from Section 3.8 for macro blocks. The performance and quality impact for macro blocks will need to be investigated.

6.1.2. Support for Carry Chains

Similar to the support for macro blocks, current academic tools do not model carry chains. Future research into carry chains will hopefully extend the framework to support such structures. The proposed solutions in Section 3.8 for carry chains can then be implemented, so that the performance and quality tradeoffs can be examined.

6.1.3. Smart Shifting

The current iPlace algorithm relies on a simplistic shifting scheme for the expansion and compaction phases. Extensions to the shifting scheme include more balanced shifting to sides with more empty CLBs. Also shifting to take into account macro-blocks and carry chains must also be examined. Changes to the shifting scheme must be considered carefully. The run-time implications must be thoroughly examined so that the speed-up is not degraded. In all, a “keep it simple” approach should be followed.

6.1.4. Analytical Placement Refinement Stage

Instead of using Simulated Annealing for the refinement phase of iPlace, other placement algorithms can also be considered to optimize the solution. One example would be the use of analytical placement algorithms such as [39] to make small changes.

6.1.5. Integration with Commercial tools

The current implementation of iPlace is tightly integrated into the TV-Pack and VPR CAD flow. It would be interesting to port and adapt iPlace into commercial frameworks such as Altera’s Quartus II framework through the Quartus University Interface Program (QUIP) package [65]. This will enable the exploration of the performance of iPlace in a commercial quality CAD environment with commercial FPGA architectures. Integration with Quartus II was attempted, but it was discovered that the Quartus CAD flow performs simultaneous clustering and placement. Imposing an external clustering constraint leads to extremely long run times. Future work is required to optimize the integration between iPlace and Quartus.

6.1.6. Incremental Routing

Placement and routing dominates the majority of the FPGA CAD flow. In order to fully complement incremental placement, incremental routing is also necessary to produce an end-to-end incremental CAD flow. For today's largest circuits, a full compilation could take an entire workday to complete. With end-to-end incremental compilation, this will hopefully reduce the compilation time necessary for incremental development.

One drawback to the iPlace approach is that it tends to shift most of the CLBs some amount. This will likely make the previous routing solution useless. Investigation into placement shifting that can co-exist with incremental routing are needed.

BIBLIOGRAPHY

[1] M. Tom, D. Leong, G. Lemieux, "Un/DoPack: Re-Clustering of Large System-on-Chip Designs with Interconnect Variation for Low-Cost FPGAs", ICCAD, 2006

[2] D. Leong, G. Lemieux, "iPlace, an incremental placement algorithm", Submitted to FPGA 2007.

[3] D. Grant, G. Lemieux, "A Semi-Synthetic Circuit Generation Technique for Testing Incremental Placement and Incremental Routing Tools", FPT, 2006.

[4] D. Grant, G. Lemieux, "`Perturber+Mutator: Semi-Synthetic Circuit Scaling for Testing Incremental Place and Route Tools", **submitted for review**, *IEEE Transactions on Computer-Aided Design*, 2006.

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

[6] D. Lewis, V. Betz, et al., "The Statix Routing and Logic Architecture", ACM International Symposium on Field-Programmable Gate Arrays, pp. 12-20, February 2002.

[7] Xilinx, Virtex-II Pro Platform FPGAs: Functional Description, ver. 2.0, June 13, 2002.

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

[9] G.Lemieux, D. Lewis, "Design of Interconnection Networks for Programmable Logic", Springer 2004

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

[11] Xilinx Virtex-V Product Brochure: http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex5/Virtex-5_LX_LXT_Product_Table.pdf

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

[13] J. Cong, 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, June 1994.

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

[15] J. Cong, and Y. Hwang, "Structural Gate Decomposition for Depth-Optimal Technology in LUT-based FPGA Designs", TODAES , Vol. 5, no. 3, July 2000.

[16] J.Cong and D.Chen, "DAOmap: A Depth-Optimal Area Optimization Mapping Algorithm for FPGA Designs", ICCAD 2004 pp 752-759

[17] K.Karplus, "Xmap: A Technology mapper for table-loopup field programmbable gate arrays", in Proc. 28th ACM/IEEE Design Automation Conference, 1991. pp 240-243

[18] R.J.Francis, J.Rose, K.Chung, "Chortle: A technology mapping program for lookup table based field programmable gate arrays", in Proc. 27th ACM/IEEE Design Automation Conf, 1990, pp 613-619

[19] R.J.Framcis, J.Rose, Z. Vranesic, "Chortle-crf: Fast technology mapping for lookup table-based FPGAs", 28th, ACI/IEEE Design Automation Conference, 1991, pp. 613-619

[20] R.Murgai, et al., "Logic synthesis algorithms for programmable gate arrays", Proc 27th ACM/IEEE Design Automation Conference, 1990, pp 620-625

[21] R.Murgai, et al., "Improved logic synthesis algorithms for table look up architectures", IEEE International Conference on Computer-Aided Design, Nov 1991, pp 564-567

[22] J. Cong, L.W. Hagen, and A.B. Kahng, "Random Walks for Circuit Clustering", IEEE Conference on Application Specific Integrated Circuits, pp. 14.2.1-14.2.4, June 1991.

[23] J. Cong and M. Smith, "A Parallel Bottom-up Clustering Algorithm with Applications to Circuits Partitioning in VLSI Design", ACM/IEEE Design Automation Conference, pp.755-60, 1993.

[24] J. Cong and S.K. Lim, "Edge Separability based Circuit Clustering with Application to Circuit Partioning", ACM/IEEE Asia South Pacific Design Automation Conference, pp. 429-434, 2000.

[25] A. Singh and M. Marek-Sadowska, "Efficient Circuit Clustering for Area and Power Reduction in FPGAs," FPGA, 2002.

[26] 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.

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

[28] 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

[29] 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.

[30] 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.

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

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

[33] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing", Science, pp. 671-680, 1983.

[34] C. Sechen and A. Sangiovanni-Vincentelli, "TimberWolf Placement and Routing Package", JSSC, pp. 510-522, 1985.

[35] M. Huang, F. Romeo, and A. Sangiovanni-Vincentelli, "An Efficient General Cooling Schedule for Simulated Annealing", International Conference on Computer Aided Design, pp. 381-384, 1986.

[36] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich, "Gordian: VLSI Placement byQuadratic Programming and Slicing Optimization", IEEE Transactions on Computer-Aided Design, pp. 356-365, 1991.

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

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

[39] K. Vorwerk, A. Kennings, A. Vannelli, "Engineering Details of a Stable force-Directed Placer", ACM/IEEE International Conference on Computer Aided Design, pp. 573-580, Nov. 7-11, 2004.

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

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

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

[43] J.Lam, JM Delosme, "Performance of a New Annealing Schedule", DAC, 1988

[44] J.S. Rose, "Parallel Golbal Routing for Standard Cells", IEEE Transactions on Computer Aided Design, pp.1085-1095, 1990.

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

[46] S. Brown, J. Rose, Z.G. Vranesic, "A Detailed Router for Field-Programmable Gate Arrays", IEEE Transactions on Computer Aided Design, pp. 620-628, 1992.

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

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

[49] M. Placzewski, "Plane Parallel A* Maze Router and Its Application to FPGAs", ACM Design Automation Conference, pp. 691-697, 1990.

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

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

[52] Y.-S. Lee, A. Wu, "A Performance and Routability Driven Router for FPGAs Considering Path Delays", ACM Design Automation Conference, pp. 557-561, 1995.

[53] J. Cong, M. Sarrafzadeh, "Incremental Physical Design," International Symposium on Physical Design, 2000, pp. 84-92.

[54] D. Singh, S. Brown, "Incremental Placement for Layout Driven Optimizations on FPGAs," ICCAD 2002, pp 752-759.

[55] P. Suaris, L. Liu et al, "Incremental Physical Resynthesis for Timing Optimization," FPGA, 2004, pp. 99-108.

[56] N. Togawa, K. Hagi, M. Yanagisawa, "An Incremental Placement and Global Routing Algorithm for Field Programmable Gate Arrays," ASP-DAC, 1998.

[57] C.S. Choy, T.S. Cheung, K.K. Wong, "Incremental layout placement modification algorithms," IEEE TCAD, April 1996, pp 437-445.

[58] J. Li, J. Yu, H. Miyashita, "An Incremental Placement Algorithm for Building Block Layout Design Based on the O_tree Non-Slicing Representation," Int'l. Conf. on Communications, Circuits and Systems, 2004. pp 1248-1252

[59] Z. Li, W. Wu, X Hong, J. Gu, "Incremental Placement Algorithm for standard Cell Layout," IEEE ISCAS, 2002, pp 883-886 Vol 2.

[60] C. Mulpuri, S. Hauck, "Runtime and Quality Tradeoffs in FPGA Placement and Routing," FPGA, 2001.

[61] Y. Sankar, J. Rose, "Trading Quality for Compile Time: Ultra-fast Placement for FPGAs," FPGA, 1999.

[62] R. Tessier, "Fast Placement Approaches for FPGAs," ACM Trans. on Design Automation of Electronic Systems, April, 2002, pp 284-305.

[63] Altera Stratix II, http://www.altera.com/products/devices/stratix2/st2-index.jsp

[64] G. Lemieux, D. Lewis, "Circuit Design of FPGA Routing Switches", FPGA, 2002

[65] Quartus University Interface Program (QUIP), http://www.altera.com/education/univ/research/unv-quip.html

[66] Saeyang Yan, "Logic Synthesis and Optimization Benchmarks User Guide, Version 3.0", MCNC, Research Triangle Park, NC, Jan 1991.

Appendix A : Single-Region Synthetic Benchmark Results

	Original	Synthetic 2.5		Synthetic 5	
	#CLB	New # CLB	ΔCLB	New # CLB	ΔCLB
alu4	153	153	3	153	9
apex2	188	188	9	188	9
apex4	127	127	3	127	9
bigkey	171	171	9	171	9
clma	839	839	25	839	49
des	160	160	9	160	9
diffeq	150	150	3	150	9
dsip	137	137	3	137	9
elliptic	361	361	9	361	25
ex5p	107	107	2	107	9
ex1010	460	460	15	460	25
frisc	356	356	9	356	25
misex3	140	140	3	140	9
pdc	458	458	15	458	25
s298	194	194	9	194	9
s38417	641	641	25	641	35
s38584	645	645	25	645	35
seq	175	175	9	175	9
spla	369	369	9	369	22
tseng	105	105	3	105	9

Table 27 Single-Region Synthetic Benchmark Circuit Statistic

	Original	Synthetic 10		Synthetic 2.5d		Synthetic 5d	
	#CLB	New # CLB	ΔCLB	New # CLB	ΔCLB	New # CLB	ΔCLB
alu4	153	153	15	153	9	175	37
apex2	188	188	25	202	23	221	58
apex4	127	127	15	127	9	127	15
bigkey	171	171	25	171	8	171	25
clma	839	839	99	899	109	967	227
des	160	160	24	173	22	192	56
diffeq	150	150	22	166	25	150	24
dsip	137	137	15	137	9	137	15
elliptic	361	361	49	361	25	361	49
ex5p	107	107	15	107	5	122	26
ex1010	460	460	46	496	61	460	46
frisc	356	356	35	356	25	356	35
misex3	140	140	15	140	9	158	33
pdc	458	458	49	490	57	525	116
s298	194	194	25	194	9	194	25
s38417	641	641	81	692	86	767	207
s38584	645	645	79	645	35	777	213
seq	175	175	25	175	9	208	57
spla	369	369	49	408	64	440	120
tseng	105	105	14	105	9	105	15

Table 28 Single-Region Synthetic Benchmark Circuit Statistic Cont'

Synthetic circuits		iPlace (inner_num=1)				
		RT (s)	CW	CP (ns)	Bbox	WL ($*10^4$)
alu4	**p25**	0.2	32	13.49	68.5	0.88
	p5	0.2	32.8	13.15	69.2	0.89
	p10	0	33	15.42	69.0	0.89
	p25d	0	32.8	12.93	69.3	0.88
	p5d	0	32	14.99	78.6	1.02
apex2	**p25**	0.4	42.4	14.34	107.4	1.46
	p5	0	42.4	14.66	107.8	1.47
	p10	0.4	42	14.20	107.7	1.46
	p25d	0.2	41.6	15.69	116.5	1.60
	p5d	0.2	42.8	17.11	126.1	1.71
apex4	**p25**	0	45	13.72	76.8	1.04
	p5	0.2	44.4	12.58	77.1	1.04
	p10	0	44.8	12.58	76.4	1.03
	p25d	0	44.8	13.42	76.7	1.04
	p5d	0	44.2	13.08	77.1	1.03
bigkey	**p25**	0.4	41.8	6.13	63.8	0.90
	p5	0.4	42.4	6.62	63.6	0.90
	p10	0.4	40.2	6.15	65.1	0.92
	p25d	0.2	42.2	6.20	63.5	0.90
	p5d	0	41	6.26	65.7	0.92
clma	**p25**	3	51.8	27.02	528.8	6.76
	p5	3	51.2	26.49	529.5	6.78
	p10	3	52.2	26.88	536.9	6.88
	p25d	3	50.6	30.91	572.7	7.23
	p5d	4	54.2	31.97	644.4	8.26

Table 29 Single-Region Synthetic Benchmark Placement Results

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		iPlace (inner_num=1)				
		RT (s)	CW	CP (ns)	Bbox	WL ($*10^4$)
des	**p25**	0	50.6	10.91	64.8	1.00
	p5	0.2	51	10.87	65.1	1.00
	p10	0	48.6	10.82	67.1	1.04
	p25d	0.2	50.6	12.35	72.5	1.10
	p5d	0	46.8	13.29	77.8	1.16
diffeq	**p25**	0.2	22	18.94	47.1	0.62
	p5	0.2	22.2	19.29	46.9	0.61
	p10	0	23.8	19.38	49.0	0.65
	p25d	0	23.2	19.48	52.1	0.67
	p5d	0	22.6	19.38	48.5	0.64
dsip	**p25**	0	37.2	6.32	45.8	0.66
	p5	0	37.2	6.38	46.0	0.66
	p10	0	37	6.35	46.2	0.67
	p25d	0.4	37.4	6.32	46.1	0.66
	p5d	0.2	37.2	6.43	46.4	0.67
elliptic	**p25**	0.2	36.4	24.47	157.3	2.06
	p5	0.4	38.2	24.96	159.7	2.06
	p10	0.4	37.6	24.45	161.8	2.10
	p25d	0	36.2	24.26	159.8	2.07
	p5d	0.6	37.6	24.97	161.0	2.09
ex5p	**p25**	0.2	44.2	14.19	63.4	0.89
	p5	0	44.8	13.12	63.6	0.90
	p10	0	43.6	15.76	63.4	0.88
	p25d	0	44	15.14	63.4	0.89
	p5d	0	43.6	14.64	70.6	0.99

Table 30 Single-Region Synthetic Benchmark Placement Results cont'

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		iPlace (inner_num=1)				
		RT (s)	CW	CP (ns)	Bbox	WL (*10^4)
ex1010	**p25**	1	47	17.82	277.8	3.66
	p5	1	46.6	16.85	276.3	3.63
	p10	1	46.4	16.81	277.3	3.64
	p25d	1.2	46.2	18.03	299.5	3.95
	p5d	1	47	17.11	279.1	3.68
frisc	**p25**	0.6	47.4	30.59	197.5	2.64
	p5	1	46.8	29.53	198.6	2.65
	p10	0.6	48.4	28.11	199.4	2.64
	p25d	1	47.6	28.87	199.5	2.64
	p5d	0.6	47.2	28.83	200.5	2.67
misex3	**p25**	0	37.4	11.39	71.1	0.94
	p5	0	37.6	13.72	71.3	0.94
	p10	0	37.6	11.73	71.2	0.94
	p25d	0	38.6	13.33	72.1	0.96
	p5d	0	37.6	13.74	81.0	1.08
pdc	**p25**	1	61.4	19.79	348.6	4.67
	p5	1.2	62	18.88	348.4	4.64
	p10	1.2	60.6	21.02	347.2	4.65
	p25d	1	61.2	25.25	367.6	4.93
	p5d	1.4	61.6	23.00	402.8	5.34
s298	**p25**	0	26.2	24.79	71.4	0.85
	p5	0	25.8	23.16	71.6	0.86
	p10	0.2	26	23.74	71.5	0.85
	p25d	0.4	25.8	23.41	71.4	0.85
	p5d	0	26.8	23.60	72.4	0.87

Table 31 Single-Region Synthetic Benchmark Placement Results cont'

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		**iPlace (inner_num=1)**				
		RT (s)	**CW**	**CP (ns)**	**Bbox**	**WL (*10^4)**
s38417	**p25**	1.2	30.4	18.16	241.7	3.00
	p5	1.8	30	17.97	239.8	2.96
	p10	1.8	31.2	18.61	243.5	3.04
	p25d	2	30.6	18.41	263.1	3.30
	p5d	2	38.8	22.81	318.1	3.89
s38584	**p25**	1	32.4	14.84	239.9	2.96
	p5	2.2	31.8	15.27	240.3	2.97
	p10	1.2	33.4	15.51	241.5	2.99
	p25d	1.6	33	15.31	241.0	2.98
	p5d	2	37.6	19.51	312.6	3.82
seq	**p25**	0	40.4	12.19	98.0	1.33
	p5	0.2	40.4	12.51	98.5	1.33
	p10	0.4	40.6	14.12	98.6	1.33
	p25d	0.4	40.8	13.29	98.6	1.33
	p5d	0.4	40.8	14.73	116.4	1.56
spla	**p25**	0.6	51.6	17.29	230.6	3.11
	p5	0.8	51.8	19.24	230.8	3.13
	p10	1	51	17.69	230.4	3.13
	p25d	0.6	49.8	19.58	250.5	3.37
	p5d	1.2	53.8	20.80	289.1	3.84
tseng	**p25**	0	21.2	16.42	31.1	0.41
	p5	0	21	16.49	30.9	0.41
	p10	0	21.4	16.42	32.3	0.43
	p25d	0	21.2	16.35	31.5	0.42
	p5d	0	21.8	16.49	32.2	0.44

Table 32 Single-Region Synthetic Benchmark Placement Results cont'

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		VPR "default" / iPlace				
		Speedup	CWQ	CPQ	BBQ	WLQ
alu4	p25	**33.00**	1.01	0.94	0.99	0.98
	p5	**35.00**	0.99	0.98	0.98	0.99
	p10	-	0.97	0.84	0.99	0.97
	p25d	-	0.99	0.98	0.99	1.00
	p5d	-	0.98	0.96	0.96	0.95
apex2	p25	**30.00**	0.98	1.00	0.99	0.99
	p5	-	0.99	0.93	0.99	0.97
	p10	**30.00**	1.00	0.98	0.98	0.99
	p25d	**67.00**	0.98	0.99	0.98	0.97
	p5d	**78.00**	0.99	0.93	0.98	0.98
apex4	p25	-	0.97	0.94	0.99	0.98
	p5	**34.00**	1.00	1.00	0.99	0.98
	p10	-	0.98	0.99	0.99	0.98
	p25d	-	1.00	1.20	0.99	0.98
	p5d	-	1.00	0.99	0.99	0.98
bigkey	p25	**42.00**	1.02	1.03	0.97	0.97
	p5	**43.00**	1.02	0.93	0.97	0.98
	p10	**45.50**	1.03	1.00	0.97	0.99
	p25d	**84.00**	0.95	0.99	0.96	0.98
	p5d	-	1.04	0.99	0.95	0.96
clma	p25	**72.00**	0.97	0.95	0.98	0.97
	p5	**70.80**	0.96	0.97	0.99	0.98
	p10	**73.53**	0.98	0.96	0.98	0.98
	p25d	**80.27**	1.00	0.89	0.98	0.97
	p5d	**69.95**	1.00	0.99	0.98	0.97

Table 33 Relative performance VPR "-default" versus iPlace

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		VPR "default" / iPlace				
		Speedup	CWQ	CPQ	BBQ	WLQ
des	p25	-	1.00	1.00	0.98	0.97
	p5	89.00	1.00	1.01	0.98	0.98
	p10	-	1.07	1.00	0.97	0.97
	p25d	102.00	0.93	0.96	0.97	0.96
	p5d	-	1.09	0.89	0.96	0.98
diffeq	p25	39.00	1.01	1.00	0.96	0.97
	p5	40.00	1.02	0.97	0.97	1.00
	p10	-	1.00	0.98	0.97	0.96
	p25d	-	0.99	0.99	0.96	0.97
	p5d	-	0.99	0.98	0.98	0.99
dsip	p25	-	1.04	0.98	0.99	0.99
	p5	-	1.05	0.98	0.98	0.99
	p10	-	1.04	0.98	0.98	0.99
	p25d	34.00	1.05	0.98	0.98	0.99
	p5d	69.00	1.04	0.95	0.98	0.98
elliptic	p25	175.00	1.03	1.26	1.00	1.00
	p5	89.00	1.00	0.95	0.99	1.01
	p10	93.50	0.99	0.98	1.00	1.00
	p25d	-	1.03	0.99	1.00	0.99
	p5d	60.67	1.01	0.96	1.00	1.00
ex5p	p25	32.00	1.00	0.91	0.99	0.98
	p5	-	0.96	1.10	0.99	0.97
	p10	-	1.01	0.82	0.99	1.00
	p25d	-	0.98	0.88	0.99	0.99
	p5d	-	0.98	0.94	0.98	0.98

Table 34 Single-Region Synthetic Benchmark Placement Results: Relative performance VPR "-default" versus iPlace Cont'

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		VPR "default" / iPlace				
		Speedup	CWQ	CPQ	BBQ	WLQ
ex1010	p25	77.60	0.99	0.91	0.99	0.99
	p5	75.00	0.99	0.95	0.99	1.00
	p10	77.00	1.03	0.97	0.99	0.99
	p25d	69.00	1.00	1.04	0.98	0.97
	p5d	76.20	1.01	0.99	0.99	0.99
frisc	p25	80.67	0.95	0.94	0.97	0.96
	p5	46.20	0.98	0.98	0.97	0.96
	p10	78.67	0.96	0.99	0.98	0.96
	p25d	47.00	0.95	0.99	0.97	0.97
	p5d	79.33	0.98	0.97	0.97	0.96
misex3	p25	-	1.01	1.17	0.99	0.98
	p5	-	0.99	0.83	0.99	0.99
	p10	-	0.99	1.09	0.99	0.99
	p25d	-	0.97	0.85	0.99	0.98
	p5d	-	0.99	1.44	0.98	0.99
pdc	p25	80.60	1.00	1.08	0.99	0.97
	p5	64.00	0.98	1.27	0.99	0.98
	p10	68.67	1.01	0.96	0.99	0.97
	p25d	84.40	0.98	0.78	0.98	0.98
	p5d	68.14	1.01	0.97	0.99	0.98
s298	p25	-	0.97	0.93	0.98	0.99
	p5	-	0.99	0.99	0.98	0.98
	p10	38.00	0.98	0.99	0.98	0.98
	p25d	19.00	0.97	1.01	0.98	0.99
	p5d	-	0.95	0.99	0.97	0.97

Table 35 Single-Region Synthetic Benchmark Placement Results: Relative performance VPR "-default" versus iPlace Cont'

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		VPR "default" / iPlace				
		Speedup	CWQ	CPQ	BBQ	WLQ
s38417	p25	**73.67**	1.00	0.94	0.99	0.99
	p5	**50.67**	1.01	0.96	0.98	0.99
	p10	**50.22**	0.99	1.02	0.98	0.97
	p25d	**49.10**	0.98	0.93	0.97	0.97
	p5d	**63.60**	0.91	0.95	0.96	0.96
s38584	p25	**104.40**	1.08	0.98	0.96	0.95
	p5	**48.55**	1.03	0.96	0.96	0.96
	p10	**90.17**	0.96	0.97	0.95	0.95
	p25d	**65.75**	1.04	0.95	0.96	0.95
	p5d	**76.90**	1.03	0.97	0.93	0.94
seq	p25	-	1.00	0.99	0.99	1.00
	p5	**61.00**	1.00	1.07	0.99	0.99
	p10	**30.50**	0.99	0.83	0.99	0.99
	p25d	**29.50**	1.00	1.01	0.99	0.99
	p5d	**36.00**	0.99	0.93	0.98	0.98
spla	p25	**75.67**	0.98	0.96	0.99	0.99
	p5	**55.50**	0.99	0.90	1.00	0.98
	p10	**44.80**	1.01	0.97	1.00	0.98
	p25d	**84.00**	1.03	0.92	1.00	1.00
	p5d	**51.17**	0.98	0.96	0.98	0.97
tseng	p25	-	1.00	1.00	0.98	0.99
	p5	-	1.01	1.00	0.98	1.00
	p10	-	1.02	1.01	0.98	0.97
	p25d	-	1.01	1.00	0.97	0.98
	p5d	-	1.00	0.99	0.97	0.97
	Geo. Mean	58.47	1.00	0.98	0.98	0.98

Table 36 Single-Region Synthetic Benchmark Placement Results: Relative performance VPR "-default" versus iPlace Cont'

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		VPR "-fast" / iPlace				
		Speedup	CWQ	CPQ	BBQ	WLQ
alu4	p25	5.00	0.99	0.95	0.99	0.99
	p5	5.00	0.99	0.98	1.00	1.01
	p10	-	0.98	0.88	0.99	1.00
	p25d	-	1.01	1.14	1.00	1.01
	p5d	-	0.99	1.03	0.98	0.97
apex2	p25	4.00	1.00	1.06	1.00	1.00
	p5	-	1.00	0.95	1.00	1.00
	p10	2.50	1.00	0.97	1.00	1.02
	p25d	8.00	1.00	1.01	0.98	0.99
	p5d	10.00	0.97	0.98	0.99	0.98
apex4	p25	-	0.97	1.03	1.00	0.99
	p5	5.00	1.00	0.97	1.00	1.00
	p10	-	0.99	1.00	1.00	0.99
	p25d	-	1.00	1.01	1.00	0.99
	p5d	-	1.02	1.03	1.01	1.00
bigkey	p25	5.00	0.97	1.02	1.02	1.02
	p5	5.00	1.07	0.93	1.02	1.02
	p10	5.00	1.10	1.02	1.02	1.02
	p25d	10.00	0.99	0.99	1.02	1.01
	p5d	-	0.99	0.99	1.01	1.02
clma	p25	8.27	1.02	0.94	1.01	1.00
	p5	7.93	1.00	0.97	1.01	1.00
	p10	8.13	1.01	0.93	1.01	0.99
	p25d	8.87	1.02	0.89	1.00	0.99
	p5d	8.15	0.99	0.96	0.99	0.98

Table 37 Single-Region Synthetic Benchmark Placement Results: Relative performance VPR "-fast" versus iPlace

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		VPR "-fast" / iPlace				
		Speedup	CWQ	CPQ	BBQ	WLQ
des	p25	-	0.96	0.99	1.03	1.02
	p5	10.00	1.00	1.00	1.02	1.01
	p10	-	1.04	1.01	1.01	1.01
	p25d	12.00	0.99	0.97	1.01	1.01
	p5d	-	0.98	0.93	1.00	1.01
diffeq	p25	5.00	1.03	1.01	0.99	0.99
	p5	4.00	1.02	0.98	1.00	1.01
	p10	-	1.03	1.00	1.01	1.00
	p25d	-	1.02	1.00	0.99	1.00
	p5d	-	1.05	1.00	1.02	1.03
dsip	p25	-	1.04	1.00	1.01	1.01
	p5	-	1.05	0.96	1.01	1.02
	p10	-	1.04	0.97	1.01	1.02
	p25d	5.00	1.03	0.97	1.01	1.02
	p5d	8.00	1.04	0.98	1.01	1.01
elliptic	p25	22.00	1.03	1.01	1.03	1.03
	p5	10.00	1.01	0.96	1.02	1.02
	p10	10.00	1.02	0.99	1.03	1.02
	p25d	-	1.05	1.01	1.03	1.04
	p5d	7.33	1.02	0.95	1.02	1.03
ex5p	p25	5.00	0.99	0.94	1.00	1.00
	p5	-	0.99	1.01	1.00	0.99
	p10	-	1.02	0.83	1.01	1.01
	p25d	-	1.00	0.86	1.00	0.99
	p5d	-	0.99	0.97	1.00	1.00

Table 38 Single-Region Synthetic Benchmark Placement Results: Relative performance VPR "-fast" versus iPlace Cont'

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		VPR "-fast" / iPlace				
		Speedup	CWQ	CPQ	BBQ	WLQ
ex1010	p25	**9.20**	1.04	0.95	1.01	1.01
	p5	**8.60**	1.03	0.95	1.00	1.01
	p10	**8.80**	1.03	0.98	1.00	1.00
	p25d	**8.17**	1.03	1.05	1.00	0.99
	p5d	**8.80**	1.01	0.95	1.01	1.00
frisc	p25	**9.00**	0.99	0.97	1.01	1.00
	p5	**5.00**	1.02	0.98	1.00	1.00
	p10	**9.00**	0.99	0.99	1.00	1.01
	p25d	**5.60**	0.97	0.97	0.99	0.99
	p5d	**9.33**	1.01	0.99	1.00	1.00
misex3	p25	**-**	1.02	1.02	1.00	1.00
	p5	**-**	1.01	0.91	1.00	1.00
	p10	**-**	1.00	0.98	1.00	1.00
	p25d	**-**	0.98	1.03	0.99	1.00
	p5d	**-**	0.99	0.97	0.99	1.00
pdc	p25	**9.80**	0.99	0.96	1.00	1.00
	p5	**7.00**	0.97	1.35	1.00	0.99
	p10	**7.00**	1.03	0.95	1.01	1.00
	p25d	**10.20**	0.99	0.76	0.99	0.98
	p5d	**7.86**	1.03	1.06	1.01	1.00
s298	p25	**-**	0.98	0.95	1.00	1.01
	p5	**-**	1.02	0.99	1.00	0.99
	p10	**5.00**	0.98	0.99	1.00	1.00
	p25d	**2.00**	1.02	0.99	1.00	1.00
	p5d	**-**	0.98	0.98	0.99	1.00

Table 39 Single-Region Synthetic Benchmark Placement Results: Relative performance VPR "-fast" versus iPlace Cont'

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		VPR "-fast" / iPlace				
		Speedup	CWQ	CPQ	BBQ	WLQ
s38417	p25	**8.83**	1.03	0.96	1.03	1.03
	p5	**5.89**	1.05	1.00	1.04	1.04
	p10	**5.56**	1.06	0.95	1.03	1.02
	p25d	**5.60**	1.03	0.93	1.03	1.02
	p5d	**7.40**	0.92	0.95	1.00	1.00
s38584	p25	**12.00**	0.98	1.00	1.00	1.00
	p5	**5.73**	1.03	1.00	1.00	1.00
	p10	**10.33**	0.96	1.01	1.00	1.00
	p25d	**7.38**	0.95	0.95	1.00	1.01
	p5d	**8.40**	1.04	0.97	0.98	0.98
seq	p25	**-**	1.00	0.97	1.00	1.01
	p5	**7.00**	1.00	0.98	1.00	1.01
	p10	**3.00**	1.00	0.84	1.00	1.01
	p25d	**4.00**	0.99	1.08	0.99	1.01
	p5d	**4.00**	1.01	0.98	0.99	0.99
spla	p25	**8.67**	1.00	1.23	1.01	1.01
	p5	**6.75**	0.99	0.91	1.01	1.00
	p10	**5.00**	1.02	1.07	1.01	1.00
	p25d	**9.67**	1.04	0.92	1.01	1.01
	p5d	**6.00**	1.01	1.04	1.00	0.99
tseng	p25	**-**	1.06	1.00	1.01	1.03
	p5	**-**	1.01	1.00	1.01	1.03
	p10	**-**	1.01	1.00	1.01	1.01
	p25d	**-**	1.04	1.01	1.01	1.02
	p5d	**-**	0.98	1.01	1.01	1.02
	Geo. Mean	**6.85**	1.01	0.98	1.00	1.01

Table 40 Single-Region Synthetic Benchmark Placement Results: Relative performance VPR "-fast" versus iPlace Cont'

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		VPR "-superfast" / iPlace				
		Speedup	CWQ	CPQ	BBQ	WLQ
alu4	**p25**	**2.00**	1.08	1.05	1.04	1.04
	p5	**1.00**	1.05	1.16	1.05	1.05
	p10	**-**	1.02	0.94	1.03	1.03
	p25d	**-**	1.04	1.05	1.04	1.06
	p5d	**-**	1.07	1.01	1.04	1.04
apex2	**p25**	**0.50**	1.05	1.06	1.05	1.07
	p5	**-**	1.06	0.97	1.05	1.07
	p10	**1.00**	1.09	1.01	1.05	1.07
	p25d	**4.00**	1.02	1.03	1.03	1.03
	p5d	**4.00**	1.07	0.98	1.05	1.06
apex4	**p25**	**-**	1.02	0.97	1.04	1.03
	p5	**2.00**	1.03	1.05	1.03	1.03
	p10	**-**	1.02	1.04	1.04	1.03
	p25d	**-**	1.02	1.01	1.04	1.04
	p5d	**-**	1.05	0.94	1.03	1.03
bigkey	**p25**	**1.00**	0.98	1.00	1.10	1.08
	p5	**1.50**	0.88	0.95	1.10	1.07
	p10	**0.50**	0.96	1.01	1.10	1.08
	p25d	**-**	0.99	1.04	1.10	1.08
	p5d	**-**	0.97	1.00	1.08	1.06
clma	**p25**	**1.73**	1.10	0.96	1.11	1.10
	p5	**1.80**	1.13	0.98	1.13	1.12
	p10	**1.73**	1.10	0.96	1.11	1.10
	p25d	**1.87**	1.11	0.90	1.09	1.09
	p5d	**2.00**	1.08	0.95	1.08	1.08

Table 41 Single-Region Synthetic Benchmark Placement Results: Relative performance VPR "-superfast" versus iPlace

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		VPR "-superfast" / iPlace				
		Speedup	CWQ	CPQ	BBQ	WLQ
des	p25	-	0.95	1.02	1.13	1.12
	p5	-	0.93	1.03	1.14	1.12
	p10	-	0.99	1.05	1.11	1.11
	p25d	2.00	0.94	0.97	1.11	1.10
	p5d	-	0.91	0.94	1.11	1.09
diffeq	p25	2.00	1.15	0.99	1.12	1.12
	p5	-	1.13	0.98	1.12	1.14
	p10	-	1.08	1.00	1.10	1.08
	p25d	-	1.16	0.98	1.11	1.10
	p5d	-	1.14	0.98	1.11	1.09
dsip	p25	-	1.04	0.96	1.11	1.10
	p5	-	1.03	0.96	1.11	1.11
	p10	-	1.01	0.98	1.10	1.09
	p25d	1.00	1.02	0.97	1.10	1.08
	p5d	1.00	1.02	0.97	1.10	1.07
elliptic	p25	6.00	1.12	1.03	1.12	1.11
	p5	3.00	1.07	1.00	1.11	1.12
	p10	2.50	1.10	1.02	1.12	1.12
	p25d	-	1.13	1.01	1.12	1.12
	p5d	2.00	1.11	1.06	1.11	1.12
ex5p	p25	1.00	1.04	0.98	1.05	1.03
	p5	-	1.03	1.13	1.05	1.03
	p10	-	1.03	0.88	1.05	1.05
	p25d	-	1.04	0.94	1.04	1.04
	p5d	-	1.01	1.04	1.05	1.05

Table 42 Single-Region Synthetic Benchmark Placement Results: Relative performance VPR "-superfast" versus iPlace Cont'

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		VPR "-superfast" / iPlace				
		Speedup	CWQ	CPQ	BBQ	WLQ
ex1010	p25	2.20	1.06	0.95	1.05	1.04
	p5	2.00	1.12	1.01	1.08	1.09
	p10	2.00	1.06	0.97	1.05	1.05
	p25d	2.00	1.09	0.96	1.07	1.06
	p5d	2.20	1.08	0.95	1.06	1.06
frisc	p25	2.67	1.07	1.00	1.09	1.09
	p5	1.80	1.09	1.00	1.09	1.08
	p10	2.00	1.07	1.02	1.10	1.11
	p25d	1.40	1.07	1.00	1.09	1.09
	p5d	2.33	1.10	0.99	1.09	1.09
misex3	p25	-	1.04	1.18	1.04	1.05
	p5	-	1.05	0.95	1.04	1.06
	p10	-	1.06	1.09	1.04	1.05
	p25d	-	1.03	1.02	1.03	1.03
	p5d	-	1.05	1.00	1.04	1.05
pdc	p25	2.20	1.07	1.06	1.06	1.06
	p5	1.83	1.07	1.35	1.07	1.07
	p10	2.00	1.09	1.00	1.08	1.06
	p25d	2.60	1.05	0.81	1.05	1.05
	p5d	1.86	1.08	0.96	1.07	1.07
s298	p25	-	1.04	0.94	1.06	1.05
	p5	-	1.07	1.01	1.05	1.04
	p10	-	1.05	0.99	1.05	1.06
	p25d	1.00	1.06	0.97	1.05	1.05
	p5d	-	1.03	0.98	1.05	1.04

Table 43 Single-Region Synthetic Benchmark Placement Results: Relative performance VPR "-superfast" versus iPlace Cont'

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic circuits		VPR "-superfast" / iPlace				
		Speedup	CWQ	CPQ	BBQ	WLQ
s38417	p25	**2.17**	1.20	0.94	1.20	1.20
	p5	**1.33**	1.21	0.96	1.24	1.23
	p10	**1.44**	1.21	0.97	1.20	1.18
	p25d	**1.50**	1.14	0.98	1.16	1.15
	p5d	**1.90**	1.07	0.98	1.19	1.18
s38584	p25	**3.20**	1.14	1.02	1.16	1.16
	p5	**1.45**	1.13	0.98	1.14	1.14
	p10	**2.33**	1.10	1.04	1.12	1.12
	p25d	**2.00**	1.14	0.98	1.16	1.15
	p5d	**1.90**	1.14	0.99	1.14	1.16
seq	p25	**-**	1.06	1.03	1.05	1.06
	p5	**2.00**	1.06	1.01	1.05	1.06
	p10	**1.00**	1.06	0.90	1.05	1.06
	p25d	**1.50**	1.07	0.97	1.06	1.08
	p5d	**1.00**	1.08	1.00	1.06	1.08
spla	p25	**2.33**	1.08	1.11	1.09	1.09
	p5	**1.50**	1.07	0.96	1.10	1.09
	p10	**1.20**	1.09	0.97	1.08	1.08
	p25d	**3.00**	1.09	1.02	1.09	1.08
	p5d	**1.50**	1.09	0.99	1.07	1.06
tseng	p25	**-**	1.08	1.02	1.11	1.12
	p5	**-**	1.08	1.03	1.11	1.12
	p10	**-**	1.08	1.01	1.10	1.09
	p25d	**-**	1.07	1.02	1.10	1.11
	p5d	**-**	1.10	1.01	1.10	1.10
	Geo. Mean	1.75	1.06	1.00	1.08	1.08

Table 44 Single-Region Synthetic Benchmark Placement Results: Relative performance VPR "-superfast" versus iPlace Cont'

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Synthetic Circuit	Syn 2.5	Syn 5	Syn 10	Syn 2.5d	Syn 5d	GeoMean
alu4	2.24	3.28	3.64	3.08	2.33	2.86
apex2	1.22	1.23	1.54	1.75	2.18	1.55
apex4	2.53	2.29	2.65	2.13	2.49	2.41
bigkey	4.75	4.60	4.13	3.92	4.91	4.45
clma	2.41	2.64	2.05	5.24	3.66	3.02
des	4.09	4.29	4.80	4.47	4.92	4.50
diffeq	2.86	2.67	2.55	3.58	2.86	2.88
dsip	2.88	2.56	2.45	2.11	2.64	2.51
elliptic	4.27	4.97	3.37	4.78	3.13	4.03
ex5p	2.38	2.54	1.86	2.02	2.68	2.27
ex1010	2.36	1.96	2.47	2.84	2.86	2.48
frisc	2.73	2.73	3.69	2.56	2.34	2.78
misex3	1.78	2.33	1.72	1.80	2.62	2.02
pdc	4.01	3.09	2.24	4.64	4.41	3.55
s298	2.47	2.05	2.35	2.71	1.73	2.23
s38417	4.36	3.94	3.22	4.55	5.30	4.22
s38584	5.02	4.89	6.27	4.59	6.40	5.38
seq	1.90	1.77	1.77	2.23	2.83	2.07
spla	3.09	3.77	4.63	3.83	5.27	4.05
tseng	2.99	3.25	3.02	3.50	2.72	3.08
					Geomean	2.97

Table 45 Single-Region Synthetic Benchmark Placement Results: Placement Stability for VPR Baseline

Synthetic Circuit	Syn 2.5	Syn 5	Syn 10	Syn 2.5d	Syn 5d	GeoMean
alu4	1.57	1.54	1.65	1.55	2.33	1.71
apex2	1.39	1.34	1.40	2.00	2.35	1.65
apex4	1.99	1.96	1.81	2.05	1.79	1.92
bigkey	1.28	1.29	1.42	1.25	1.51	1.35
clma	2.73	2.69	2.72	3.19	3.98	3.02
des	1.23	1.15	1.50	2.31	2.84	1.69
diffeq	1.53	1.47	1.76	2.33	1.60	1.71
dsip	0.99	1.11	1.23	1.13	1.17	1.12
elliptic	2.28	2.41	2.40	2.36	2.37	2.36
ex5p	1.55	1.55	1.31	1.57	2.59	1.66
ex1010	1.53	1.60	1.52	1.88	1.50	1.60
frisc	2.84	2.80	2.82	2.92	2.83	2.84
misex3	1.35	1.47	1.47	1.50	2.14	1.57
pdc	2.11	2.11	2.14	2.91	3.05	2.43
s298	2.15	2.32	2.39	2.33	2.36	2.31
s38417	2.15	2.16	2.43	2.81	3.69	2.59
s38584	2.46	2.40	2.60	2.36	4.75	2.80
seq	1.61	1.66	1.67	1.53	2.80	1.81
spla	1.66	1.69	1.87	2.30	2.91	2.04
tseng	1.05	1.10	1.12	1.25	1.42	1.18
					Geomean	1.89

Table 46 Single-Region Synthetic Benchmark Placement Results: Placement Stability for iPlace

Appendix B: Single-Region Physical Re-Synthesis Benchmark Results

Single-Region Physical Resynthesis Benchmark Circuit	iPlace (inner_num=1)				
	RT (s)	CW	CP (ns)	Bbox	WL $*10^4$
clma - 2.5	3.0	49.4	26.2	528	6.83
clma - 5	3.4	48.6	27.9	529	6.76
clma - 10	3.0	49.4	26.8	535	6.83
clma - 15	3.4	50.6	26.6	546	6.99
ex1010 - 2.5	1.2	46.4	16.2	278	3.66
ex1010 - 5	1.0	46.8	16.5	281	3.71
ex1010 - 10	1.6	46.2	16.6	283	3.71
ex1010 - 15	1.2	45.0	16.8	284	3.71
misex3 - 5	-	37.6	12.3	73	0.95
misex3 - 15	0.2	35.8	12.7	75	1.01
pdc - 2.5	1.2	60.4	22.3	349	4.65
pdc - 5	1.0	61.0	19.3	352	4.67
pdc - 10	1.0	60.2	21.1	353	4.67
pdc - 15	1.0	59.8	19.9	356	4.71
spla - 2.5	0.2	50.6	16.4	231	3.10
spla - 5	0.4	51.0	17.4	234	3.14
spla - 10	0.4	51.0	17.4	235	3.15
spla - 15	1.0	50.2	16.7	239	3.19
Geo. Mean	1.0	49.5	18.9	282	3.72

Table 47 Single-Region Physical Resynthesis Benchmark iPlace Placement Results

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Single-Region Physical Resynthesis Benchmark Circuit	VPR "default" / iPlace				
	Speedup	CWQ	CPQ	BBQ	WLQ
clma - 2.5	**70.3**	0.97	1.00	0.98	0.97
clma - 5	**64.8**	1.00	0.91	0.98	0.98
clma - 10	**76.9**	1.00	0.93	0.98	0.98
clma - 15	**67.9**	0.96	0.96	0.98	0.97
ex1010 - 2.5	**59.3**	1.00	0.99	0.99	0.99
ex1010 - 5	**74.4**	0.99	1.09	0.98	0.98
ex1010 - 10	**46.8**	1.00	1.00	0.99	0.98
ex1010 - 15	**62.8**	1.00	0.99	0.99	0.98
misex3 - 5	**-**	0.98	0.97	0.99	1.00
misex3 - 15	**36.0**	0.97	1.00	0.97	0.96
pdc - 2.5	**63.3**	0.99	0.84	0.98	0.97
pdc - 5	**80.2**	0.98	1.01	0.98	0.98
pdc - 10	**69.2**	0.98	0.96	0.98	0.97
pdc - 15	**70.8**	0.99	0.94	0.99	0.98
spla - 2.5	**257.0**	1.01	1.02	1.00	0.99
spla - 5	**106.5**	0.98	1.02	0.99	0.97
spla - 10	**109.5**	0.99	1.22	0.99	0.98
spla - 15	**44.8**	0.98	0.99	0.99	0.98
Geo. Mean	**71.9**	0.99	0.99	0.99	0.98

Table 48 Single-Region Physical Resynthesis Benchmark Relative Performance, VPR default versus iPlace

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Single-Region Physical Resynthesis Benchmark Circuit	VPR "-fast" / iPlace				
	Speedup	CWQ	CPQ	BBQ	WLQ
clma - 2.5	**7.6**	1.01	0.97	1.00	0.98
clma - 5	**7.4**	1.02	0.89	1.01	1.01
clma - 10	**8.4**	1.03	0.93	1.01	1.01
clma - 15	**7.8**	0.98	1.03	1.01	0.99
ex1010 - 2.5	**6.7**	1.02	1.04	1.00	0.99
ex1010 - 5	**8.4**	1.00	1.03	1.00	0.99
ex1010 - 10	**5.4**	1.00	1.02	1.00	1.00
ex1010 - 15	**7.2**	1.02	0.99	1.00	1.00
misex3 - 5	**-**	0.99	0.93	0.99	1.00
misex3 - 15	**4.0**	0.97	0.92	0.98	0.97
pdc - 2.5	**6.8**	1.00	0.84	1.00	0.99
pdc - 5	**8.4**	1.00	1.03	0.99	0.99
pdc - 10	**7.8**	1.00	0.88	1.00	0.99
pdc - 15	**8.2**	1.00	0.94	0.99	0.99
spla - 2.5	**28.0**	1.03	1.09	1.01	1.00
spla - 5	**12.0**	1.01	1.02	1.01	0.99
spla - 10	**14.5**	1.00	1.07	1.01	1.00
spla - 15	**5.2**	1.01	1.04	1.00	0.99
Geo. Mean	8.1	1.01	0.98	1.00	0.99

Table 49 Single-Region Physical Resynthesis Benchmark Relative Performance, VPR "-fast"versus iPlace

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Single-Region Physical Resynthesis Benchmark Circuit	VPR "-superfast" / iPlace				
	Speedup	CWQ	CPQ	BBQ	WLQ
clma - 2.5	**1.7**	1.08	0.96	1.09	1.08
clma - 5	**1.7**	1.14	0.95	1.12	1.11
clma - 10	**1.9**	1.12	0.96	1.11	1.11
clma - 15	**1.5**	1.08	1.06	1.11	1.09
ex1010 - 2.5	**1.7**	1.11	1.02	1.07	1.07
ex1010 - 5	**2.0**	1.07	1.09	1.07	1.06
ex1010 - 10	**1.3**	1.08	1.02	1.06	1.06
ex1010 - 15	**1.7**	1.12	1.02	1.08	1.08
misex3 - 5	**-**	1.04	1.08	1.03	1.04
misex3 - 15	**-**	1.03	1.03	1.02	1.02
pdc - 2.5	**1.8**	1.09	0.85	1.06	1.06
pdc - 5	**2.0**	1.07	1.14	1.06	1.07
pdc - 10	**2.0**	1.07	1.03	1.06	1.07
pdc - 15	**2.2**	1.06	1.16	1.05	1.05
spla - 2.5	**6.0**	1.08	1.13	1.07	1.07
spla - 5	**2.5**	1.05	0.99	1.07	1.06
spla - 10	**3.0**	1.07	1.05	1.08	1.07
spla - 15	**1.6**	1.06	1.23	1.08	1.08
Geo. Mean	2.0	1.08	1.04	1.07	1.07

Table 50 Single-Region Physical Resynthesis Benchmark Relative Performance, VPR "-super-fast" versus iPlace

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Appendix C: Multi-Region Physical Re-Synthesis Benchmark Results

Multi Region Physical-Resynthesis Benchmark Circuit	iPlace(inner_num=1)				
	RT (s)	CW	CP (ns)	Bbox $*10^3$	WL $*10^5$
clone - 50	72.2	111.0	72.4	4.32	5.30
clone - 40	57.2	110.0	72.5	3.90	4.87
clone - 30	64.8	114.2	72.1	3.87	4.85
clone - 20	57.6	117.4	71.2	3.79	4.77
clone - 10	58.0	112.8	71.9	3.90	4.88
stdev0 - 50	60.8	92.6	74.4	4.27	5.22
stdev0 - 40	63.0	90.6	72.4	4.20	5.15
stdev0 - 30	76.2	92.0	74.2	4.26	5.21
stdev0 - 20	71.0	95.6	74.1	3.91	4.85
stdev0 - 10	59.8	93.6	73.0	4.00	4.96
stdev010 - 50	89.0	140.2	75.8	4.23	5.26
stdev010 - 40	66.0	140.0	74.3	4.04	5.08
stdev010 - 30	63.8	142.0	75.0	4.03	5.07
stdev010 - 20	56.4	150.6	74.0	3.93	4.98
stdev010 - 10	70.4	144.4	74.3	4.01	5.05
Geo. Mean	65.2	114.6	73.4	4.04	5.03

Table 51 Multi Region Physical Re-Synthesis Benchmark iPlace Placement Results

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Multi Region Physical-Resynthesis Benchmark Circuit	VPR "default" / iPlace				
	Speedup	CWQ	CPQ	BBQ	WLQ
clone - 50	**62.8**	0.92	0.99	0.92	0.92
clone - 40	**70.0**	0.97	1.02	0.96	0.96
clone - 30	**68.3**	0.96	1.00	0.96	0.96
clone - 20	**61.5**	0.98	0.98	0.96	0.96
clone - 10	**75.9**	0.98	0.98	0.95	0.95
stdev0 - 50	**67.3**	0.92	0.95	0.94	0.95
stdev0 - 40	**66.8**	0.97	0.98	0.97	0.97
stdev0 - 30	**55.1**	0.96	0.97	0.96	0.96
stdev0 - 20	**56.0**	0.96	0.95	0.95	0.96
stdev0 - 10	**66.3**	0.96	0.96	0.95	0.95
stdev010 - 50	**47.1**	0.97	0.96	0.96	0.96
stdev010 - 40	**59.7**	0.98	0.97	0.96	0.96
stdev010 - 30	**68.6**	0.98	0.98	0.96	0.97
stdev010 - 20	**66.5**	0.99	0.98	0.97	0.96
stdev010 - 10	**58.5**	0.97	0.97	0.95	0.96
Geo. Mean	63.0	0.96	0.98	0.96	0.96

Table 52 Multi Region Physical Re-Synthesis Benchmark Relative Performance, VPR default versus iPlace

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Multi Region Physical-Resynthesis Benchmark Circuit	VPR "-fast" / iPlace				
	Speedup	CWQ	CPQ	BBQ	WLQ
clone - 50	6.7	0.96	0.98	0.96	0.96
clone - 40	7.9	1.00	0.98	0.99	0.99
clone - 30	7.5	1.00	0.99	0.99	0.98
clone - 20	6.6	1.00	1.00	0.99	0.99
clone - 10	8.2	1.01	0.99	0.99	0.98
stdev0 - 50	7.6	1.00	0.98	0.99	0.98
stdev0 - 40	6.9	1.03	1.00	1.01	1.00
stdev0 - 30	5.6	0.98	0.97	0.98	0.98
stdev0 - 20	6.5	1.00	0.95	0.99	0.99
stdev0 - 10	7.1	1.00	0.96	0.99	0.99
stdev010 - 50	4.9	1.00	0.99	1.00	0.99
stdev010 - 40	6.3	0.99	0.98	0.98	0.98
stdev010 - 30	8.0	1.02	1.00	0.99	0.99
stdev010 - 20	7.1	1.00	0.98	0.99	0.99
stdev010 - 10	5.8	1.00	0.99	0.99	0.98
Geo. Mean	6.8	1.00	0.98	0.99	0.99

Table 53 Multi Region Physical Re-Synthesis Benchmark Relative Performance, VPR "-fast" versus iPlace

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality

Multi Region Physical-Resynthesis Benchmark Circuit	VPR "-superfast" / iPlace				
	Speedup	CWQ	CPQ	BBQ	WLQ
clone - 50	**1.2**	1.11	1.03	1.11	1.09
clone - 40	**1.5**	1.11	1.04	1.11	1.10
clone - 30	**1.4**	1.15	1.03	1.14	1.12
clone - 20	**1.5**	1.10	1.03	1.14	1.11
clone - 10	**1.3**	1.12	1.00	1.11	1.09
stdev0 - 50	**1.5**	1.14	0.98	1.10	1.08
stdev0 - 40	**1.4**	1.21	1.03	1.14	1.11
stdev0 - 30	**1.1**	1.23	1.01	1.13	1.11
stdev0 - 20	**1.1**	1.19	0.97	1.13	1.11
stdev0 - 10	**1.4**	1.17	1.00	1.12	1.10
stdev010 - 50	**0.9**	1.13	0.98	1.15	1.13
stdev010 - 40	**1.3**	1.14	1.00	1.14	1.12
stdev010 - 30	**1.7**	1.10	0.99	1.11	1.10
stdev010 - 20	**1.4**	1.11	0.99	1.12	1.11
stdev010 - 10	**1.2**	1.10	0.99	1.11	1.10
Geo. Mean	**1.3**	1.14	1.00	1.12	1.11

Table 54 Multi Region Physical Re-Synthesis Benchmark Relative Performance, VPR "-super-fast" versus iPlace

Legend:	
-	Results with Run time too fast to measure
Bbox	Total bounding box
BBQ	Bounding Box Quality
CP	Critical Path
CPQ	Critical Path Quality
CW	Channel Width
CWQ	Channel Width Quality
Quality	Experimental result relative to iPlace
RT	Average Run time of the simulation
Speedup	Speed up in placement time
WL	Total WireLength
WLQ	WireLength Quality