國 立 交 通 大 學

電信工程學系

碩 士 論 文

具備最小干擾於標準元件下的電路快速合法化方法

FastLegalize: Legalization for Standard Cell Based Design with Minimal Disturbance

研 究 生：吳宗祐

指導教授：李育民 教授

中 華 民 國 九 十 八 年 七 月

具備最小干擾於標準元件下的電路快速合法化方法
FastLegalize: Legalization for Standard Cell Based Design with Minimal Disturbance

研 究 生：吳宗祐　　　　Student：Tsung-You Wu

指導教授：李育民　　　　Advisor：Yu-Min Lee

國 立 交 通 大 學
電 信 工 程 系
碩 士 論 文

A Thesis
Submitted to Department of Communications Engineering
College of Electrical and Computer Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Communications Engineering

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

具備最小干擾於標準元件下的電路快速合法化方法

學生:吳宗祐　　　　　　　　　　　　　　　　　　指導教授:李育民 博士

國立交通大學電信工程學系碩士班

摘　　要

隨著製程進入奈米時代，現今的積體電路中有著數以百萬計的可移動標準元件(standard cell)與固定單元(fixed macros)。在做電路合法化時為了維持原有的全域擺置結果，在晶片上標準元件的移動量必須要最小化。因此在這一篇論文中我們提出了一個有效將電路中標準元件移動量達到最小化的方法。

為了能夠有效限制標準元件的移動量，首先將晶片切割成每一個大小都相同的格子，然後從密度最大的格子開始做合法化。由於密度太大的格子裡的空間不足給標準元件做合法化，因此必須嘗試合併周圍的格子直到它的密度小於臨界值，在這裏我們提出兩種結構以有效的去合法化電路。

合併後的格子開始透過我們有效的方法去做合法化，所提出的方法可同時維持在全域擺置結果的特性與最小化移動距離。為了能夠改進效能，在合法化的過程中每個標準元件都會隨時做擺置上的更新。直到所有的格子都做完合法化後即結束。

與最新的研究結果“Abacus”比較後，我們所提出的方法可減少平均48%

的移動量，最大的移動量可減少140%，此外，執行上的速度有將近1.11倍的提升。實驗結果證明我們的方法可以獲得一個好的合法化電路。

FastLegalize: Legalization for Standard Cell Based Design with Minimal Disturbance

Student: Tsung-You Wu Advisor: Dr. Yu-Min Lee

Department of Communications Engineering
National Chiao Tung University

ABSTRACT

An efficient legalization approach is necessary for the integrated circuit design which consists of millions of movable standard cells and fixed macros. To maintain the global placement result, the disturbance of cells must be minimized. In this work, a fast legalization placer, *FastLegalize*, is developed to legalize standard cells with minimal movement.

First, a chip is divided into several bins with equal size to limit the movable scope of cells. Then, starting with the most crowed unlegalized bin, a merging procedure for bins is used to integrate bins into a cross-shaped bin-merged structure or a square-shaped bin-merged structure until the cell density in that bin-merged structure is less than a defined threshold. After that, an efficient legalization method which simultaneously preserves the ordering in each row and minimizes the sum of absolute movement distances is developed to legalize cells in that bin-merged structure. To improve the legalization quality, the proposed legalization method refreshes the positions of legalized cells during legalization. Finally, the above legalizing procedure is repeated until all cells are non-overlapped.

Compared with the state-of-the-art method, Abacus, *FastLegalize* can reduce the total movement of cells to be 48% in average, and save the largest movement of cells to be 140% in average. Moreover, *FastLegalize* can obtain average 1.11X runtime speed up.

誌　謝

這篇論文能順利完成，首先要由衷感謝我的指導老師李育民博士，老師豐富的研究經驗引導徬徨無助的我一條光明之路，使得這篇論文能夠順利完成。在這研究所兩年生涯中，老師所給予我的不僅研究還有正確的人生觀，使得在未來的人生旅途中不至於走偏，這對我未來就業或者是進修有著相當大的影響的。在交大研究所這兩年我成長許多，處理事情的態度、對於一件事的熱忱、判斷一件事情的正確性都有相當程度的改變。現在的我能夠抬頭挺胸走出國立交通大學，這一切都要感謝我的指導教授與交大所帶給我的一切。

關於實驗的進行，感謝臺灣大學電子研究所張耀文老師實驗室所提供的全域合法化工具。如果沒有張老師的幫忙，實驗的進行將會變得相當困難，在此對張老師致上深深的謝意。

感謝實驗室的領航員：培育學長與柏毅學長。我永遠不會忘記培育學長在半夜跟我一同討論數學式子推導，您對數學的那種熱忱使我在數學拾起了信心；感謝柏毅學長總是抽空幫我解決我的難題，您的幫助對我在研究方法有著另類的見解。感謝焯基學長，您在生活、研究、把妹都對我幫助良多，雖然您總是如此的嘴砲，但幫助學弟的心卻是如此的熱血，我很慶幸在兩年半前與您美麗邂逅，彩繪了我的研究生涯。感謝學長斯安，您酷酷的表情讓我學習到如何當真男人。謝

謝同儕懷中，您對研究的認真引領我向您看齊，您傑出的表現隱約告訴我該認真研究不能再玩了。感謝實驗室學弟志升與學妹們亭蓉、書函、巧翎，您的陪伴讓我在研究的路上不孤單。感謝成功高中的安祺老師與交大英教所的阿偉幫我修正論文裡的英文詞句，你們的教導使我在英文寫作中進步許多。還有交大所有陪伴著我的學長姐、同學、學弟妹們，感謝你們豐富了我的研究生涯。深深感謝你們！

特別感謝我的父母親，從小到大你們無怨無悔對我付出栽培才能使得我能夠順利完成碩士學業，你們對我的恩情無法三言兩語述說的。謹在此將本論文文獻獻給你們，共享這份喜悅與榮耀。

吳宗祐

撰於 國立交通大學

2009/07/31

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The thesis is mainly to present novel techniques for the legalization problem, which is the second stage of the placement in physical design. As the process technology scales down, the design becomes more complex dramatically. There are new challenges to handle the legalization problem in placement such as minimization of movement and wirelength. An inferior legalization will make design failure since the design does not meet the constraints produced by global placement stage. In this thesis, we propose an efficient legalization approach to handle these challenges.

1.1 Design Flow of an Integrated Circuit

In Very Large Scale Integration (VLSI) design flow, an VLSI chip is designed from a system specification, next following a series of procedures, and packaged eventually. A classical VLSI design flow is shown in Figure 1.1. We focus on the legalization, which is the placement stage of physical design procedure. Before the physical design procedure, the logic circuit design procedure is usually described in a circuit diagram which contains the circuit elements (*e.g.,* transistors and blocks) and interconnections also called a netlist between components. Moreover, a netlist is normally created from a logic description by utilizing logic synthesis tools. Because of the great complexity, the chip is partitioned into sub-chips which are known as function blocks in the floorplanning step. Then, the best location of each circuit component and/or pin locations according to appropriate cost functions in a chip are determined in the placement stage. According to the netlist, the interconnects between components will be completed and consume minimum cost in routing. Finally, the circuit must be verified such that all

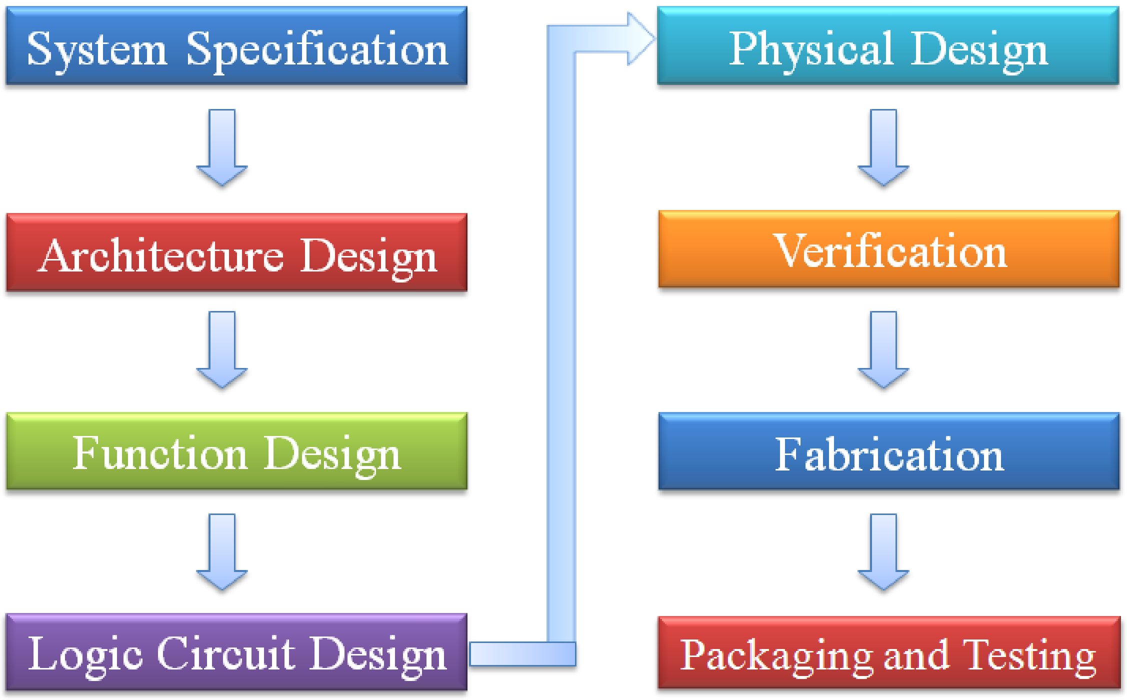

Figure 1.1: The VLSI design flow.

constraints are met and packaged in a chip. Otherwise, the circuit must be checked and fixed for the problems in each step.

1.2 Motivation

As process technology scales into the sub-micron and nano-meter region, the placement task for large chips has become more important than before [3]. The rapid growth of the complexity of VLSI circuits has further pushed the requirement of efficient and fast automatic placer. In general, the designed placer is divided into three stages for different considerations: global placement, legalization, and detailed placement stages. Different stage of the placer considers the different factors. Several approaches exist for global placement [4–6]. The global placement result is legalized then. To maintain the various characteristics produced by global placement result, such as wirelength, routing congestion, timing optimization, and heat distribution, movement of total cells in circuits should be minimized so as to disturb the original global placement result as little as possible in legalization. In most research of the placement, the legalization is seldom discussed specifically; nevertheless, it plays increasingly important roles today.

In modern circuits, there are millions of cells and blocks in a chip, as shown in Figure 1.2. It

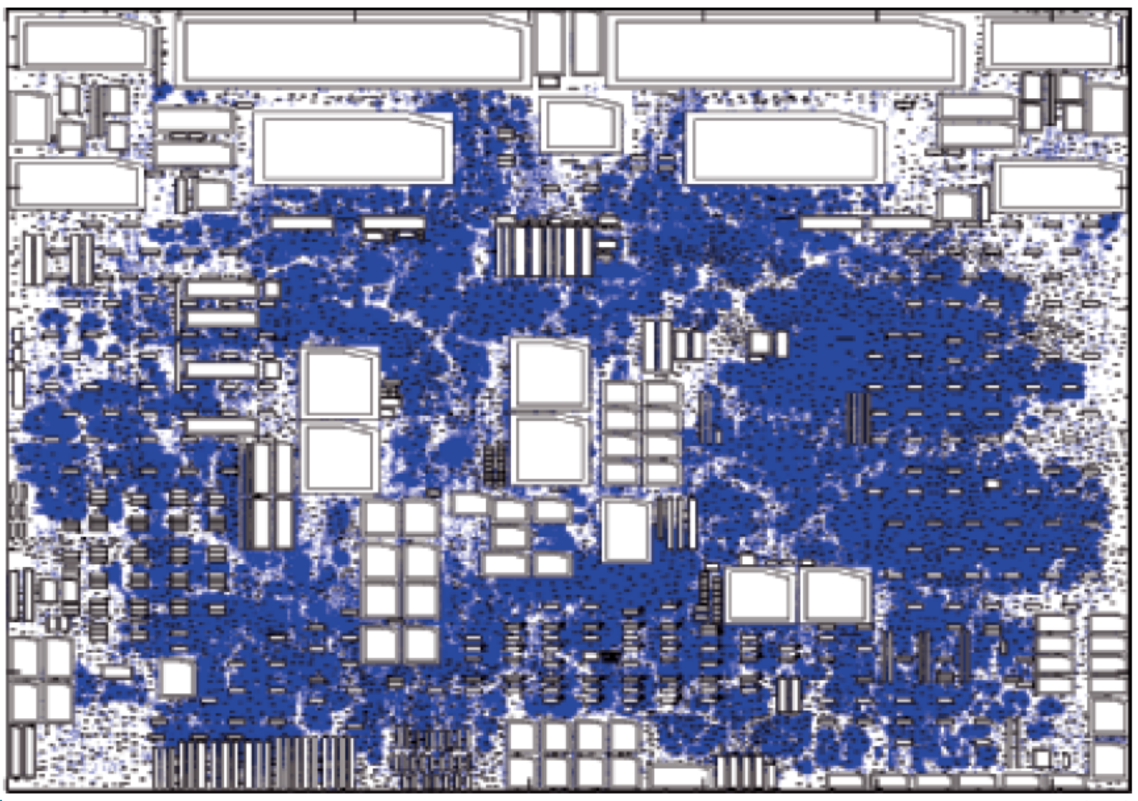

Figure 1.2: Placement circuit.

is important to design an efficient legalization placer; hence, our goal is to propose an efficient and fast legalization approach to solve the complex problems.

1.3 Our Features

In this work, we develop an efficient legalization approach, *FastLegalize*, and it has the following distinguished features.

- Bin Merged Structure: We propose two structures of integrated bins that can help us limit the movable scope for each cell.
- Legalization Framework: The approach places the cells into one row which is the lowest cost among all rows; moreover, it preserves the relative order of cells in each row.
- Legalization Core (LegCore): It refreshes the positions of legalized cells during legalization; i.e., the approach preserves the best position for cells instantly.
- Sum of Absolute Distances Solver (SADS): The solver minimizes the Manhattan dis-

tances of total cells between global placement and legalization; moreover, it uses the median concept for getting the optimal solution quickly. It is more exact than estimation approaches proposed before.

Tetris [7] is a well-known patent for legalization. It is used popularly by the integrated placer (including all stages of placement), such as NTUplace3 [4] and DPlace 2.0 [6]. Recently, Abacus [2] uses dynamic programming to improve the performance such that it has smaller movement than Tetris. Experimental results show that *FastLegalize* can reduce the total movement of cells to be 109% in average, and save the largest movement of cells to be 104% in average. Moreover, *FastLegalize* can obtain average 3.30× runtime speed up. Besides, our algorithm preserves the relative order of cells in each row.

1.4 Organization of the Thesis

The rest of this thesis is organized as follows. Chapter 2 introduces the basic concept of placement. At the same time, we also review some related literatures. In chapter 3, we present the Bin Merged Procedure, Legalization Framework, Legalization Core (LegCore), and Sum of Absolute Distances Solver (SADS), which are the kernel algorithms of this thesis. The experimental results of the comparison for the proposed *FastLegalize* with Abacus [2] are presented in chapter 4. Finally, the conclusion is given in chapter 5.

Chapter 2

Preliminary and Literature Overview

2.1 Physical Design

Physical design is one of the most important design procedures because it impacts the power, area, and performance of electronic circuits. It is a flow that converts a system description into a geometric circuit; moreover, it mainly contains the following tasks: partitioning, floorplanning, placement, and routing, as shown in Figure 2.1.

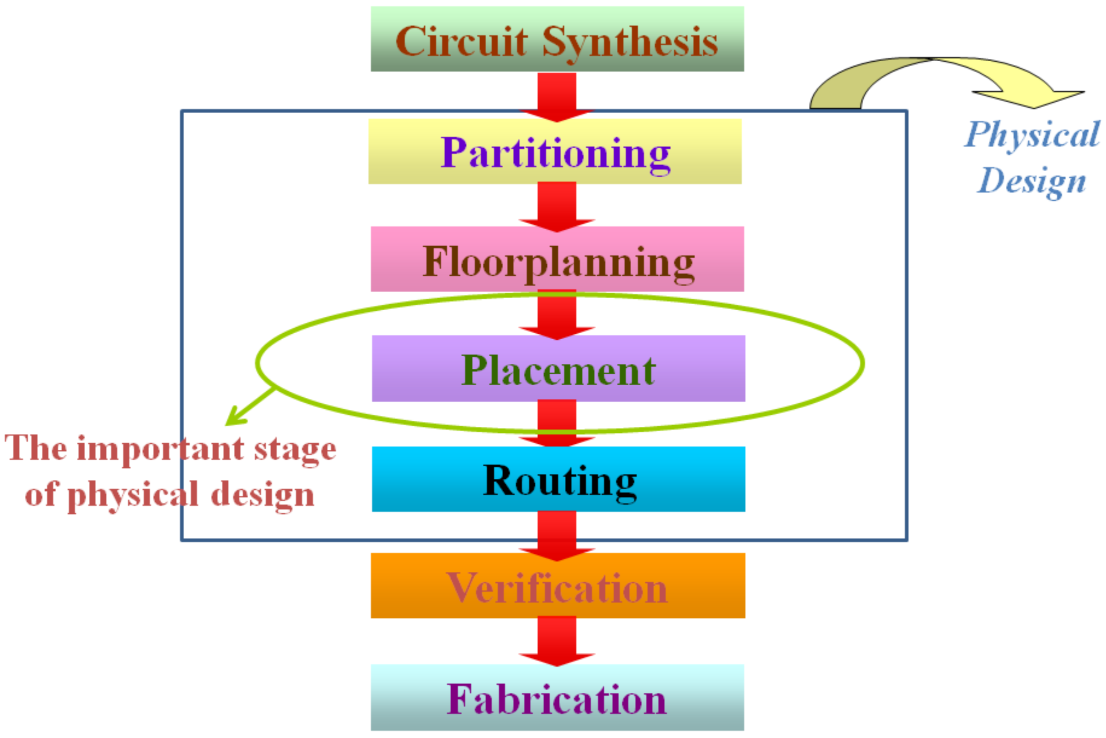

Figure 2.1: The main tasks in the physical design flow, partitioning, floorplanning, placement, and routing.

A circuit assigned in the chip is divided into several parts by partitioner such that they are manageable and the number of connection among the parts is minimized. Floorplanning makes enveloped rectangle rooms which are non-overlapping divided by horizontal and vertical line segments; furthermore, floorplanning optimizes area, wirelength, and etc.. Placement determines the best position of each circuit component (*e.g.,* standard cells and blocks) in a chip. Given different constraints, router tries to route all nets completely.

Among all tasks in the physical design flow, placement is the most critical part on the quality of final design, especially for modern design circuits. Without a high-quality placement which takes all constraints into consideration, the routing will be failed because the nets are not to be connected completely. Consequently, it is important to focus on the study of placement problems.

2.2 Basic Concept of Placement

Placement is a complex problem to analysis and solve; therefore, it is usually divided into three parts, global placement, legalization, and detailed placement. Different stage of the placement considers the different factors. They are described as follows.

Global placement: *Global placement places the modules roughly into the placement region that may violate some placement constraints (e.g., the modules may be overlapped) while maintaining some characteristics (e.g., wirelength, routing congestion, timing optimization, and heat distribution). An example of the global placement result is shown in Figure 2.2(a).*

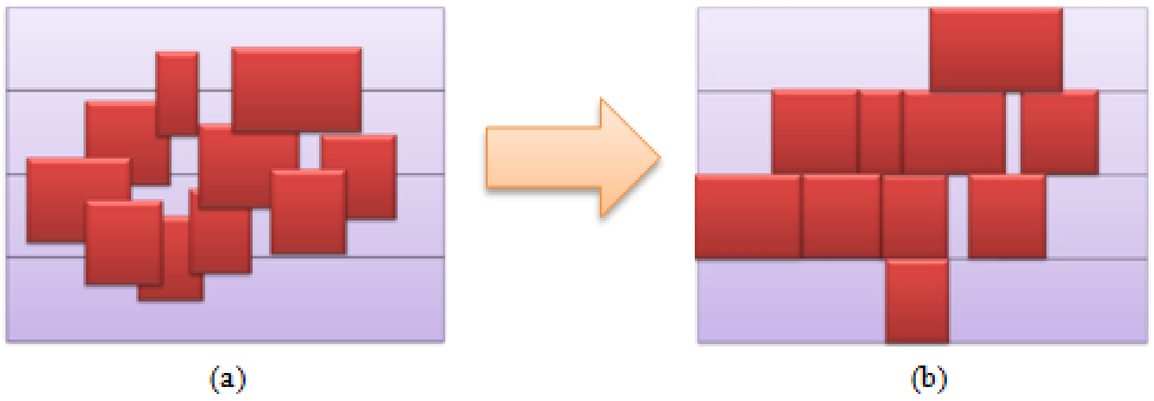

Figure 2.2: The placement procedure. (a) Global placement result generated by a rough placement solution. (b) Legalization result generated by moving modules to be legal from global placement solution.

Legalization: *Legalization is to move the modules from their global placement locations to legal placement site with minimum total movement and non-overlaps among them. An example* *of the legalization result is shown in Figure 2.2(b).*

Detailed Placement: *Detailed placement further improves some placement objective of legalization solution such as wirelength or congestion, and produces a better legal placement solution.*

Among all steps in placement, legalization has the significant impact on the final placement result. The final placement with some constraints produced by global placement stage will be failed if the constraints are not met after legalization stage. If the cells are not placed well, the design even can not be fabricated. Hence, we focus on the study of legalization problems.

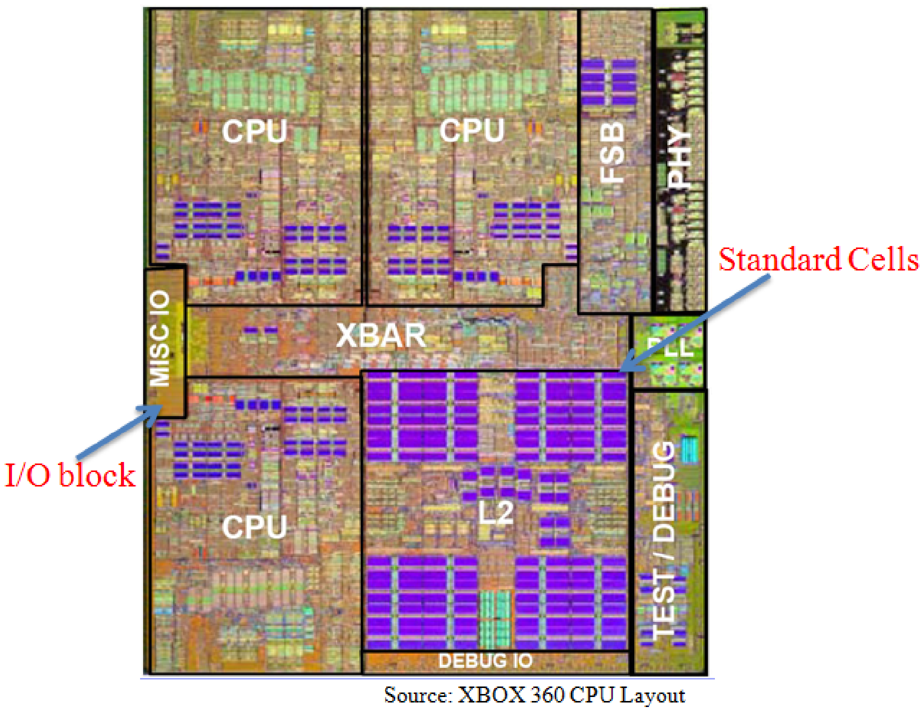

Figure 2.3: Cells and blocks in a chip

2.2.1 Standard Cells

There are millions of standard cells which construct the functions in a chip. A standard cell consists of transistors and interconnects, and provides a boolean logic function (*e.g.,* AND, OR, inverters, and etc..) or a storage function (flipflop or latch). Though different in width, the standard cells have the same height. Moreover, the construction of cells is the same if their number of input/output pins and functions are the same. Consequently, it can reduce the complexity of design. The physical representation in a chip is shown in Figure 2.3.

2.2.2 Fixed Blocks

In a circuit, some important functions are pre-designed and placed at fixed position before placement stage for taking some factors into consideration, such as I/O object, memory, cache, and etc.. Since the fixed blocks slice the rows into pieces of rows, it makes the legalization more complicated to be solved than before.

2.3 Cost Function to Legalization

In the legalization stage of placement procedure, legalization approach moves the cells overlap-free to the rows with minimal movement such that all constraints are preserved from global placement result. In general, there are two well-known methods to estimate the cost during legalization approach, Euclidean movement and Manhattan distance, used to measure the movement of all movable cells between global placement and legalization, geometrically shown in Figure 2.4.

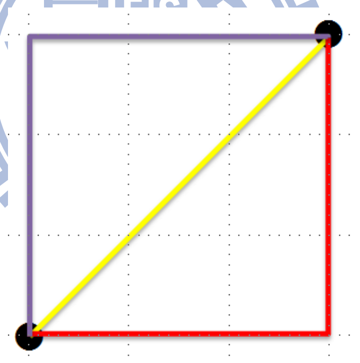

Figure 2.4: Manhattan distance plotted in red or purple line versus Euclidean movement plotted in yellow line.

Generally, the Euclidean movement between points $P = (p_1, p_2, \cdots, p_m)$ and $Q = (q_1, q_2, \cdots, q_m)$, in Euclidean m-space, is defined as:

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_m - q_m)^2} = \sqrt{\sum_{i=1}^{m} (p_i - q_i)^2} \tag{2.1}$$

It can be rewritten since the legalization is a two dimension plane problem, $P = (p_x, p_y)$ representing the global placement position and $Q = (q_x, q_y)$ representing the legalization position, computed as:

$$\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \tag{2.2}$$

(2.2) represents the movement of one cell; however, the total movement of n cells is to be calculated with minimal movement during legalization in a chip. Hence, the cost function is derived as follows.

$$\sum_{i=1}^{n} \sqrt{(p_x^i - q_x^i)^2 + (p_y^i - q_y^i)^2} \tag{2.3}$$

Additionally, the Manhattan distance is the summation of the lengths of the projections of the line segment between the two points, global placement and legalization position onto the coordinate axes. For example, in the plane of a chip, the Manhattan distance between the the global placement position P with coordinates (p_x, p_y) and legalization position (q_x, q_y) is $|p_x - q_x| + |p_y - q_y|$ for one cell. The total movement of n cells is to be calculated with minimal movement during legalization in a chip. Hence, the cost function is derived as follows.

$$\sum_{i=1}^{n} (|p_x^i - q_x^i| + |p_y^i - q_y^i|) \tag{2.4}$$

In this thesis, we use the Manhattan distance as our representation of cost function.

2.4 Literature Survey of Legalization

There are various approaches existing for legalizing standard cell circuit. A well-known legalization approach using heuristic cell movement is Tetris [7], used popularly by the integral placer, such as NTUPlace3 [4], DPlace 2.0 [6], and others [8–10]. Tetris proposed by D. Hill sorts the cells into a processing order based on the respective x-coordinate positions of the cells, and then places the cells one by one from the order in the row of lowest cost. The legalization of one cell is done by moving the cell over free rows. The movement is done until the nearest free row is found. Once a cell has been legalized, it will not be moved anymore. The [9, 11, 12] extends the method by Tetris which can process the objects with uniform height only; it allows the legalization placer with processing both standard cells and macros. Abacus [2] proposed by Spindler et. al. uses a Tetris-like method but it moves the legalized cells by dynamic programming. Besides, it uses a quadratic programming based technique to find a row of lowest cost for minimizing the movement from the original global placement result. The average movement is about 30 % lower in Abacus than in Tetris.

The diffusion based migration technique [13, 14] formulates the legalization as physical diffusion problem. They propose a mathematical model to model the physical diffusion. First, the chip is divided into several equal size bins, and the cells are moved to the position calculated by the mathematical model then. The authors of [15–17] presented a traditional method simulated annealing to legalize the circuits, and they starts from an initial placement with temperature. The algorithm [18] proposed by Min Pan have three stage approach: global swap is to help find the optimal region for cells, and then use vertical swap to find the good candidate space; finally, use local re-ordering to reduce the wirelength. The network flow technique [19, 20] minimizes the total movement of cells by minimum cost flow. The single row optimization technique [21, 22] places the cells in each row optimally by dynamic programming. Fractional Cut [23] divides the legalization into two stage: the cells are placed in rows by minimizing the assignment cost for assigned row, and then cells are sorted by x-coordinate and packed. The [24] presents two-phase approach: The row capacities must be met for place in, and then overlaps among modules in all rows are removed. In legalization, Mongrel [25] presents two-phase approach: The cells tries to be placed the placement region and produces a new legal placement, and then adopts the wirelengh minimization solution; hence, the constraint violation (*e.g.,* bin overflows) must be fixed by disturbing cells as little as possible. Domino [26] places the cells by solving min-cost-max-flow, and then swaps cells for reducing wirelength under non-overlap.

Furthermore, the authors of [21] indicates that initial cell ordering for standard cell circuits tested by industry tools are too good to improve it. Hence, our tool, *FastLegalize*, also preserves the initial cell ordering for preserving a good performance.

Chapter 3

The Legalization Approach for Standard Cell Based Design with Minimal Disturbance

In this chapter, we present the legalization approach for standard cell based design with minimal disturbance, also called *FastLegalize*. First, the overview of this algorithm is introduced. Then, we will present the kernel components of this algorithm in detail.

3.1 Legalization

Legalization is to place the standard cells non-overlap into the rows. Nowadays, there are millions of cells in a chip and the legalization approach of high quality is necessary. Here, we propose a hierarchical legalization placer which is fast and efficient to legalize the overlapped cells. The input of legalization is a global placement result whose cells are spread on the chip roughly and which takes some factors into consideration, such as wirelength, routing congestion, timing optimization, and heat distribution. Therefore, the legalization placer must disturb the global placement result as little as possible so as to preserve the original characteristic.

In addition, the layout of basic logic gate of standard cell is pre-designed to construct some specific functions, such as inverter, NAND, NOR, etc. Though different in width, the standard cells have the same height. In general, the rows has been divided into new subrows by macros such that they are not blocked anymore. Hence, in the rest of this thesis, we don't take the macros into consideration.

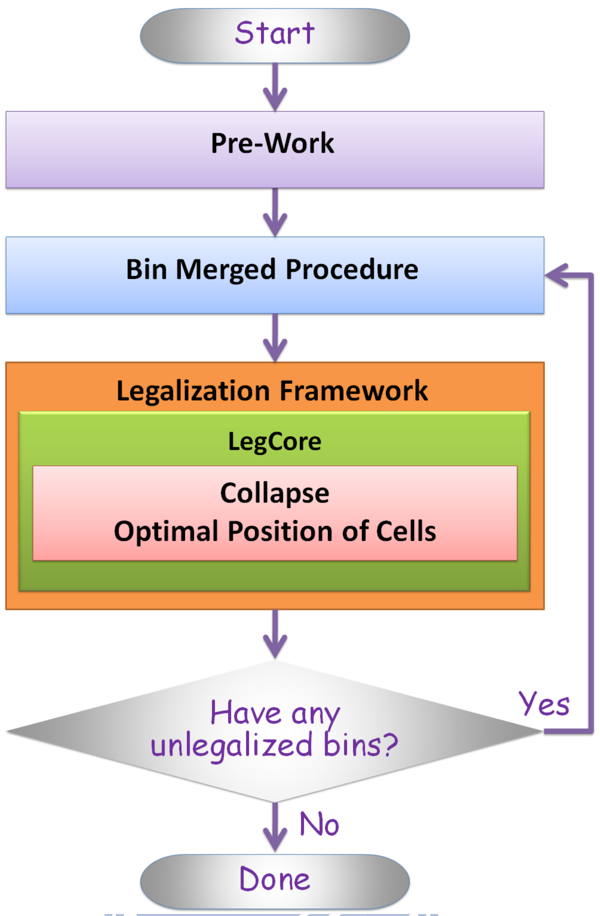

Figure 3.1: The flow chart of *FastLegalize*.

3.2 Overview of the Approach

FastLegalize essentially consists of four components: Bin Merged Procedure, Legalization Framework, LegCore, and SADS. It is targeted on reducing the movement of all standard cells under the constraints produced by global placement.

In order to reduce the movement of cells, first, we construct a bin-merged structure which is based on the cell density by Bin Merged Procedure from crowed unlegalized bins, used to limit the movable scope of cells. Moreover, the structure is constructed by integration of different bins. Then, the Legalization Framework is called by Bin Merged Procedure since the cell density of bin-merged structure is lower than defined threshold; moreover, it places the cells in a row of bin-merged structure and position of lowest cost.

To put the matter simply, the position and row for cells are calculated by LegCore in one of steps of Legalization Framework. Its kernel mainly includes two parts: Collapse and Optimal Position of Cells. The calculational solver used in Collapse is SADS, using median concept ap-

```
Bin Merged Procedure
Input:
    A chip design and global placement result
Output:
    CSBMS or SSBMS for Legalization Framework

01  Choose one unlegalized bin in the sorted order;
02  while its cell density exceeds the threshold
03      Integrate the neighbor of the bin to CSBMS;
04      if its cell density in CSBMS exceeds the threshold then
05          Integrate the corner bins of CSBMS to construct SSBMS;
06          if the cell density in SSMBS exceeds the threshold then
07              The bins in SSMBS will be merged into one bin;
08          end
09      end
10  end
```

Figure 3.2: The pseudo code of Bin Merged Procedure.

proach rather than linear programming for speed up. With those activities, our optimal position can be obtained.

Figure 3.1 is the flow chart of our algorithm, *FastLegalize*. At the beginning, the adaptable bin-merged structure is chosen by Bin Merged Procedure. Then, it enters the Legalization Framework. After that, it will enter the Bin Merged Procedure since there are any unlegalized bins in the chip. The legalization is done as bins are all legalized.

3.3 Pre-Work and Bin Merged Procedure

Some work have to do before entering the Bin Merged Procedure for required information, and it is summarized in Pre-Work Step. First, the placement is divided into bins with equal size, as shown in Figure 3.3. Based on the result obtained from the global placement, the cell density in each bin (D_i) is then analyzed. D_i is defined as the total area of all standard cells over total area of all subrows inside bin i. Then, the bins are sorted into a processing order according to descending cell density itself. Hence, the Pre-Work before Bin Merged Procedure is complete.

The pseudo code of Bin Merged Procedure is described in Figure 3.2. The standard cells are moved around the placement region based upon the bin where they are and its current cell

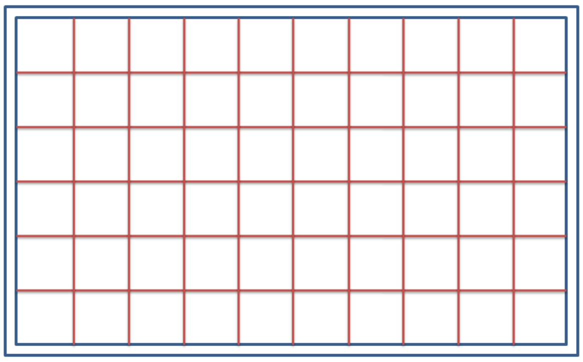
Figure 3.3: Bins with equal size

density (line 1-10). Some cells in one bin with a higher cell density will be moved away for reducing its cell density and solving the overlap among themselves; moreover, our approach preserves the relative order of the cells in each subrow, described as follows.

The unlegalized bins are processed one by one from the sorted order (line 1). For each processed bin, it is integrated with its neighbor bins to construct the Cross-Shaped Bin-Merged Structure (CSBMS), defined in the Definition 1, if the cell density exceeds the defined threshold (about 1) (line 2), and then the cell density of this bin-merged structure is calculated. If the cell density of CSBMS exceeds the threshold (line 4), the CSBMS is integrated with its corner bins to construct the Square-Shaped Bin-Merged Structure (SSBMS), defined in the Definition 2 (line 5). Otherwise, the cells in CSBMS, whose cell density is lower than the threshold, will be legalized in the next step of Legalization Framework which preserves the relative order of the cells in each subrow. Since the cell density of SSBMS still exceeds the threshold (line 6), the bins in SSBMS will be merged into one bin (line 7). Otherwise, the procedure will go to the next step of Legalization Framework. Similarly described as above, the bin merged is integrated with its neighbor bins to construct the CSBMS. It becomes a loop to integrate or merge bins until the cell density of bin-merged structure is lower than the threshold. After all bins exceeding the threshold in the sort order are processed, the remaining bins under overflowed are processed individually (line 1); hence, the legalization is done.

Definition 1 (Cross-Shaped Bin-Merged Structure, CSBMS) *The bins are integrated with its neighbor bins (the bins of up, down, left, and right bin) for reducing its density. Nevertheless, the shape of this structure is like a cross, and it is called as a Cross-Shaped Bin-Merged Structure.*

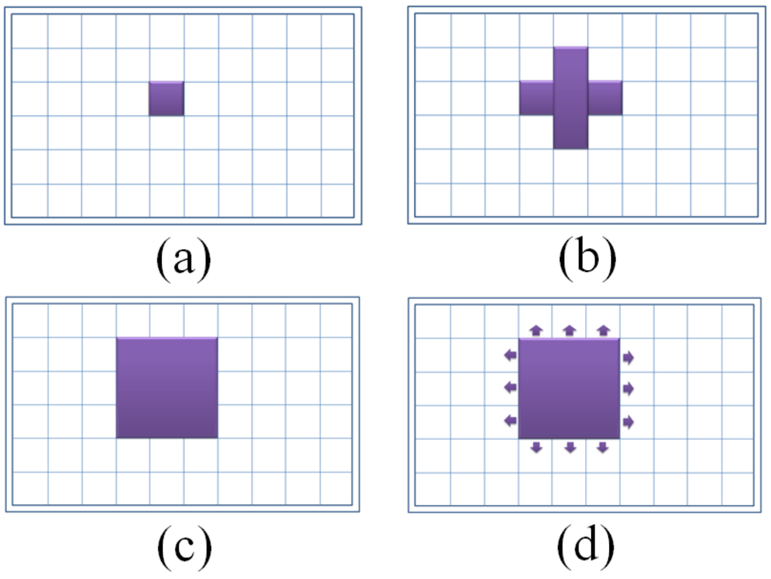

Figure 3.4: The merging procedure. (a) The most crowed unlegalized bin is chosen. (b) The neighbor bins are integrated to construct CSBMS. (c) The corner bins of CSBMS are integrated to construct SSBMS. (d) The bins in SSBMS are merged into one bin, and the merged bin is ready integrated with neighbor bins .

Definition 2 (Square-Shaped Bin-Merged Structure, SSBMS) *The bin structure, CSBMS, is integrated with its corner bins (the CSBMS of left-top, right-top, left-down, and right-down bin) for reducing its density. Nevertheless, the shape of this structure is like a square, and it is called as a Square-Shaped Bin-Merged Structure.*

In this section, two bin structures are proposed, CSBMS and SSBMS, having two advantages for legalization. One is that the movable scope of cells is limited in the CSBMS or SSBMS structure, which reduce the movement of cells efficiently. The other is that the cost time of legalization is much lower since the chip is divided into bins so as to reduce the complexity.

```
Legalization Framework
Input:
   Positions of cells before legalization in the bin structure
Output:
   Positions of cells after legalization in the bin structure

01 Sort the cells into a processing order according to ascending $x$-coordinate positions;
02 for each $cell\ i$ in the sort order do
03    $c_{best} \leftarrow \infty$;
04    $pointer \leftarrow$ the nearest subrow $r$ for $cell\ i$;
05    for the subrows around the $pointer\ r$ do
06       Insert $cell\ i$ into $subrow\ r$;
07       LegCore $r$ ($trial$);
08       Determine cost $c$;
09       if $c < c_{best}$ then $c_{best} = c, r_{best} = r$;
10       Remove $cell\ i$ from $subrow\ r$;
11    end
12 Insert $cell\ i$ into $subrow\ r_{best}$;
13 LegCore $r_{best}$ ($real$);
14 end
```

Figure 3.5: The pseudo code of Legalization Framework.

Bin Merged Procedure for CSBMS/SSBMS Graph

As described above, CSBMS/SSBMS is used for limiting the movable scope of cells. Starting with the most crowed unlegalized bin shown in Figure 3.4(a), a merging procedure is used to integrate with its neighbor bins to construct the CSBMS, shown in Figure 3.4(b). If it is still too crowed to have enough space to legalize the bin structure, it is integrated with its corner bins to construct SSMBS, shown in Figure 3.4(c). However, if the cell density of SSMBS exceeds the threshold, the integrated bins are merged into one bin. Iteratively similar steps described above as shown in Figure 3.4(d), it goes to the next step until the cell density is lower than the threshold.

3.4 Legalization Framework

After the previous algorithm, we will focus on how to find a candidate subrow for each cell in the bin-merged structure. The pseudo code of Legalization Framework is described in Figure 3.5. First, the cells are sorted into a processing order according to ascending x-coordinate positions (line 1). The cells will be legalized one by one from the processing order (line 2-14). For each cell i, it is moved over all subrows in the bin-merged structure to find a lowest cost subrow (line 5-11) in trial mode. The cost of the new position of cell i is determined (line 8) by "LegCore" (line 7) procedure (see the "LegCore" section for an overview). In detail, the cost is calculated by Manhattan distance between its global placement position and legalization position of current subrow. Finally, the cell i is removed from the current subrow (line 10) in trial mode.

The "for" loop lets each cell be moved around the nearest subrows for cell i (line 5) in the bin-merged structure, and the best candidate subrow is determined by lowest cost to cell (line 9). In the period of being moved around the subrows in bin-merged structure (line 5-11) for cell i, it is called "trial mode." The cells through LegCore (trial) procedure are not really moved to these position; nevertheless, it is moved to temporary position for getting the cost. Consequently, the best subrow needs to be placed again (line 13), and the cells through LegCore (real) procedure are actually moved to these final legal position.

$x(i), y(i)$:	lower-left coordinate of i-th cell in global placement
$x'(i), y'(i)$:	lower-left coordinate of i-th cell in legalization
$w(i)$:	width of i-th cell
N_r :	total number of ordered standard cells in one subrow
N_c^i :	total number of ordered standard cells in i-th cluster
c_r^i :	i-th cluster in the subrow r
$x_c(i)$:	optimal position of lower-left corner of i-th cluster
$w_c(i)$:	width of i-th cluster
x_{min} :	left boundary of subrow r
x_{max} :	right boundary of subrow r
B_{min} :	left boundary of the bin structure
B_{max} :	right boundary of the bin structure

Figure 3.6: Notation used in this thesis.

3.5 Legalization Core (LegCore)

LegCore is the abbreviation of "Legalization Core". It is used to solve the overlap among cells and determine the optimal positions of cells in each subrow. The algorithm of this section is divided into four parts. One is the main algorithm of the LegCore, which contains the initialization of new inserted cell, decides whether the cell is overlapped with its predecessor cluster or not, and integrates with the other two parts. Another is Collapse, which uses the SADS proposed in Section 3.6 to calculate the optimal positions for clusters by solving (3.18). Another is Optimal Position of Cells, which solves the optimal position of all cells in the last cluster of the subrow by solving (3.15). The other is cost function, which calculates the amount of influence on other cells as a new cell inserted by cell movement. Hence, the four parts are described as follows.

The notation used in the rest of this thesis is summarized in Figure 3.6. Let $x_c(i)$, $w_c(i)$ be the optimal position of lower-left coordinate and the width of i-th cluster, respectively. If the optimal position is not in the scope between x_{min} and x_{max}, which are left and right boundary of the subrow, respectively, it will be limited. Similarly, if the optimal position is not in the scope between B_{min} and B_{max}, which are left and right boundary of the bin structure, it will be limited. The algorithm of LegCore is described in Figure 3.7. First, a new inserted cell is

```
Algorithm: Legalization Core (LegCore)
Input:
   Cell $i$ in the bin-merged structure
Output:
   Cells (including cell $i$) with optimal position in the subrow $r$

01  Create a new cluster $c_r^{last}$, and it is the last cluster of subrow $r$;
02  Cell $i$ is stored in cluster $c_r^{last}$ & $x_c(c_r^{last}) \leftarrow x(i), w_c(c_r^{last}) \leftarrow w(i)$;
03  $c' \leftarrow$ Predecessor of $c_r^{last}$;
04  if $c'$ exists and $x_c(c') + w_c(c') \geq x_c(c_r^{last})$ then
05      $w_c(c') \leftarrow w_c(c') + w_c(c_r^{last})$;
06      Store all cells from cluster $c_r^{last}$ into cluster $c'$;
07      Remove cluster $c_r^{last}$;
08      Collapse $(c')$;
09  end
10  Optimal Position of Cells();
11  Cost Function();
```

Figure 3.7: Algorithm of Legalization Core (LegCore).

initialized before legalization (line 1-2). In initialization, the cell is included in a new cluster whose optimal position and width are the same as the cell, respectively (line 2). Moreover, a cluster can include lots of cells but only one cell is included in it initially. The cluster is processed according to ascending x-coordinate positions; therefore, early processed cluster is absolutely left of lately processed cluster. It means, the relative order of the cells in the legal placement is the same in the global placement. If the last cluster c_r^{last} is overlapped with the predecessor cluster c', the two clusters will be merged into one cluster whose cells in cluster c_r^{last} and c' are included and the Collapse is called to calculate the position of the merged cluster (line 4-9). Nevertheless, the subCollapse of Collapse is called continually as long as the new position of cluster is overlapped with its predecessor cluster until the overlap among all clusters is solved (subCluster of Collapse is called by itself; see the overview in Section 3.5.1). Finally, Optimal Position of Cells is called to calculate the optimal position of all cells included in the last cluster in the subrow (line 10).

The LegCore we presented is partly similar to [2, 21, 22]. The cells are placed by dynamic refresh or theirs with preserving the order of cells and no overlap. Because of the optimality proof presented in [21], it can state that our LegCore is also optimal.

```
Algorithm: Collapse
Input:
    The cluster $c_r^{last}$
Output:
    The overlap is solved among clusters

01  Calculate $\Delta x_j = x(j) - \sum_{k=1}^{j-1} w(k)$ for the last cell $j$ of cluster $c_r^{last}$;
02  Sort the sorted sequence $\{\Delta x_j\}_{j=1}^{(i-1)}$ of $c_r^{last}$ and $\Delta x_1$ into $\{\Delta x'_j\}_{j=1}^{i}$ by insertion sort;
03  $Q_{1/2}(\{\Delta x'_j\}_{j=1}^{i})$ is obtained from (3.18);
04  $x_c(c_r^{last}) \leftarrow Q_{1/2}(\{\Delta x'_j\}_{j=1}^{i})$;
    \\Limit $x_c(c_r^{last})$ position between $x_{min}$ and $x_{max} - w_c(c_r^{last})$;
05  if $x_c(c_r^{last}) < x_{min}$ then $x_c(c_r^{last}) = x_{min}$;
06  if $x_c(c_r^{last}) > x_{max} - w_c(c_r^{last})$ then $x_{max} - w_c(c_r^{last})$;
    \\Limit $x_c(c_r^{last})$ position between $B_{min}$ and $B_{max} - w_c(c_r^{last})$;
07  if $x_c(c_r^{last}) < B_{min}$ then $x_c(c_r^{last}) = B_{min}$;
08  if $x_c(c_r^{last}) > B_{max} - w_c(c_r^{last})$ then $B_{max} - w_c(c_r^{last})$;
09  $c' \leftarrow$ Predecessor of $c_r^{last}$;
    \\Do $c'$ and $c_r^{last}$ overlap with each other?
10  if $c'$ exists and $x_c(c') + w_c(c') > x_c(c_r^{last})$ then
11      $w_c(c') \leftarrow w_c(c') + w_c(c_r^{last})$;
12      Store all cells from cluster $c_r^{last}$ into cluster $c'$;
13      subCollapse($c', c_r^{last}$)
14  end
```

Figure 3.8: Algorithm of Collapse.

```
Algorithm: subCollapse
Input:
   Two clusters $c'$ and $c_r^{last}$
Output:
   The overlap is solved among clusters

01 Calculate $\Delta x_j = \Delta x_j - \sum_{k=1}^{n} w(k)$ of cluster $c'$ for each cell $j$ in $c_r^{last}$;
02 Sort the sequence $\{\Delta x_j\}_{j=1}^{i}$ of $c'$ and $\{\Delta x_j\}_{j=1}^{k}$ of $c_r^{last}$ into $\{\Delta x'_j\}_{j=1}^{(i+k)}$ by merging sort;
03 $Q_{1/2}(\{\Delta x'_j\}_{j=1}^{(i+k)})$ is obtained from (3.18);
04 $x_c(c') \leftarrow Q_{1/2}(\{\Delta x'_j\}_{j=1}^{(i+k)})$;
   \\Limit $x_c(c')$ position between $x_{min}$ and $x_{max} - w_c(c')$;
05 if $x_c(c') < x_{min}$ then $x_c(c') = x_{min}$;
06 if $x_c(c') > x_{max} - w_c(c')$ then $x_{max} - w_c(c')$;
   \\Limit $x_c(c')$ position between $B_{min}$ and $B_{max} - w_c(c')$;
07 if $x_c(c') < B_{min}$ then $x_c(c') = B_{min}$;
08 if $x_c(c') > B_{max} - w_c(c')$ then $B_{max} - w_c(c')$;
09 Remove cluster $c_r^{last}$ and $c'' \leftarrow$ Predecessor of $c'$;
   \\Do $c''$ and $c'$ overlap with each other?
10 if $c''$ exists and $x_c(c'') + w_c(c'') > x_c(c')$ then
11    $w_c(c'') \leftarrow w_c(c'') + w_c(c')$;
12    Store all cells from cluster $c'$ into cluster $c''$;
13    subCollapse($c''$, $c'$)
14 end
```

Figure 3.9: Algorithm of subCollapse.

3.5.1 Collapse

The algorithm of Collapse is described in Figure 3.8, and subCollapse called by Collapse is described in Figure 3.9. The optimum position is obtained from the procedure line 1-4 of Collapse or subCollapse, which will be described in Section 3.6. After the optimal position of cluster c is obtained, the position must be checked whether it is in the scope of subrow or not. The left corner $x_c(c)$ of cluster c is limited to right of x_{min} and B_{min}, and the right corner of $x_c(c) + w_c(c)$ is limited to left of x_{max} and B_{max} (line 5-8) of Collapse or subCollapse. If the new position of cluster is overlapped with its predecessor cluster (line 10) of Collapse or subCollapse, the width and the cells of the cluster is added to the predecessor cluster for merging (line 11-13). Then, the subCollapse is called to legalize between them (line 13). It is called continually until the overlap among clusters are solved.

```
Algorithm: Optimal Position of Cells
Input:
    Cells of the last cluster in the subrow $r$
Output:
    Cells with optimal position of the last cluster in the subrow $r$

1  $c_r^{last} \leftarrow$ the last cluster $c$ in the subrow $r$;
2  $x'(1)$ of cluster $c_r^{last} \leftarrow x_c(c_r^{last})$;
3  for cell $i = 2, \cdots, N_c^{last}$ in the cluster $c_r^{last}$ do
4      $x'(i) = x'(1) + \sum_{k=1}^{i-1} w(k)$;
5  end
```

Figure 3.10: Algorithm of Optimal Position of Cells.

3.5.2 Optimal Position of Cells

The algorithm of Optimal Position of Cells is described in Figure 3.10. After all overlaps are solved among the clusters in the subrow described in previous subsections, the positions of cells in the last cluster are determined in line 2-5. The position of first cell is determined by the position of the cluster (line 2), and the position of second cell is $x'(1) + w(1)$ because it is abutted with the first cell and so on (line 4).

The main difference to Abacus is that our SADS minimizes the amount of total cell movement. Because of the median concept manipulated by SADS described in Section 3.6, lots of constraints are not solved necessarily. Additionally, our approach relies on simple data structure; complex data structure is not necessary in our approach.

3.5.3 Cost Function

As a new cell is inserted and placed in subrow r, the Cost function calculates the influence on affected cells and determines the cost. The cost includes two parts, the difference of x- and y-coordinate in distance between global placement and legalization of all affected cells. Figure 3.11 gives an example to illustrate the legalization procedure.

Figure 3.11(a) is one bin of unlegalized circuit. In Figure 3.11(b), cell 1 is legalized in optimal position, and cell 2 is moved to the nearest row in y-direction. There is an overlap

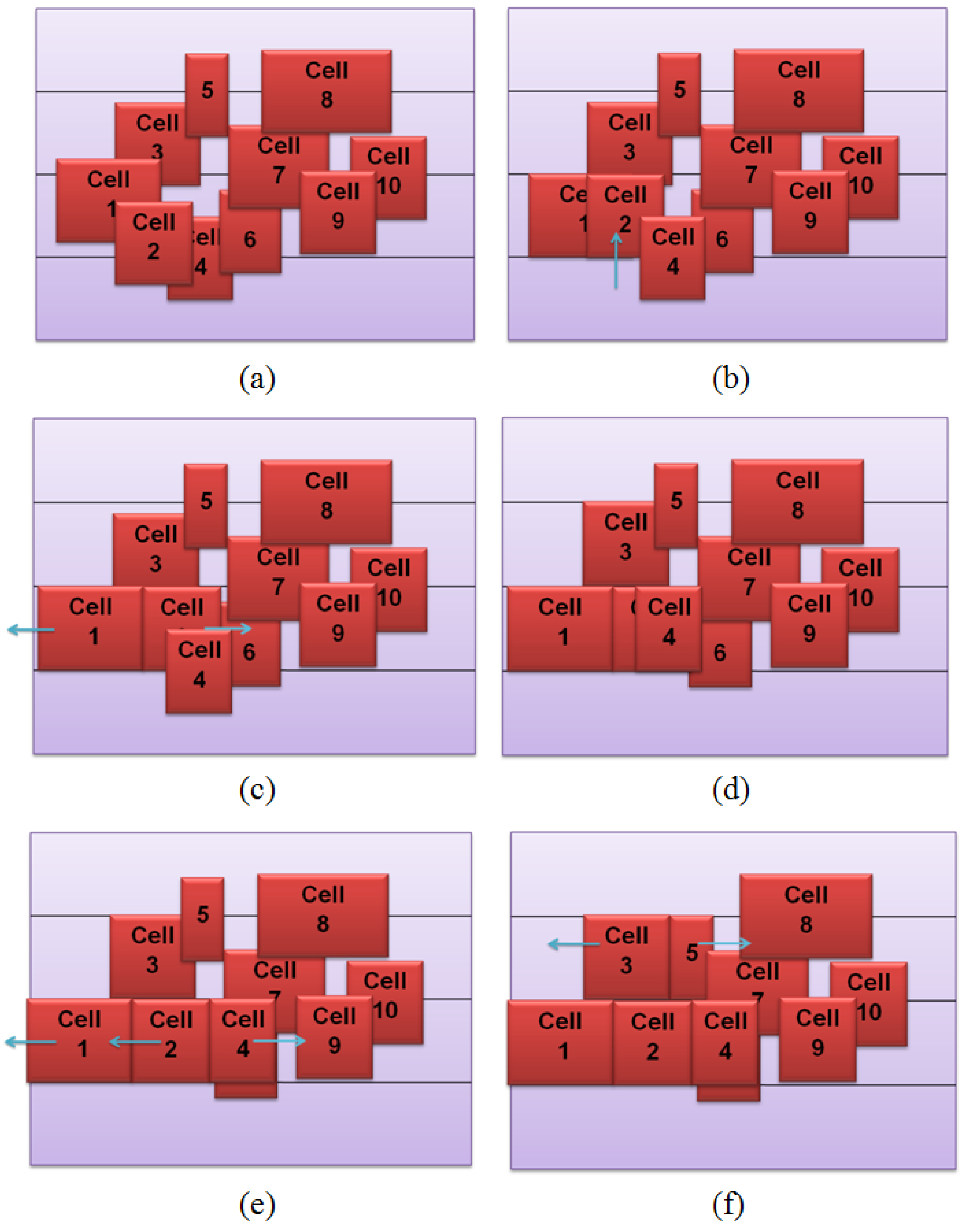

Figure 3.11: An example for illustrating the legalization procedure.

between cell 1 and 2, and the LegCore is called to solve the overlap between them. The overlap between them is solved, as shown in Figure 3.11(c). At this time, the cost is calculated and includes the movement of cell 1 and cell 2 between global placement and legalization position. Similarly shown in Figure 3.11(d), cell 2 have to be moved as cell 4 is overlapped with cell 2. Note that the cell 1 has to be moved since cell 1 and cell 2 are in the same cluster; hence, Figure 3.11(e) shows the legalized cell 1, 2, and 4, and the cost of three cells are included. Similarly shown in Figure 3.11(f), the cost of cell 3 and 5 are included as the cell 3 is affected and cell 5 is inserted.

3.6 The Optimal Position for Clusters by Sum of Absolute Distances Solver (SADS)

Let N_r be the total number of ordered standard cells along x-direction in one subrow, indexed from 1 to N_r, and let $x(l)$ and $y(l)$ be the lower-left corner x- and y-coordinate of cell l in the global placement, respectively, and $w(l)$ is the width of cell l .

To minimize the disturbance, the problem is formulated to minimize the Manhattan distance movement.

$$\min \left\|\mathbf{x}'_{N_r} - \mathbf{x}_{N_r}\right\|_1 + \left\|\mathbf{y}'_{N_r} - \mathbf{y}_{N_r}\right\|_1, \tag{3.1}$$

where

$$\begin{aligned}
\mathbf{x}'_{N_r} &= \left(x'(1), \cdots, x'(l), \cdots, x'(N_r)\right)^T, \\
\mathbf{y}'_{N_r} &= \left(y'(1), \cdots, y'(l), \cdots, y'(N_r)\right)^T, \\
\mathbf{x}_{N_r} &= \left(x(1), \cdots, x(l), \cdots, x(N_r)\right)^T, \\
\mathbf{y}_{N_r} &= \left(y(1), \cdots, y(l), \cdots, y(N_r)\right)^T.
\end{aligned}$$

Here, $x'(l)$ and $y'(l)$ are the x- and y-coordinate positions of the l-th cell after legalization, respectively.

Our approach here is to minimize the disturbance as cells are aligned overlap-free to a subrow with keeping the original ordering along x-direction as the y-coordinate position of each cell is fixed. Since each $y'(l)$ is fixed, the problem of (3.1) is equivalent to

$$\begin{aligned} &\min \left\|\mathbf{x}'_{N_r} - \mathbf{x}_{N_r}\right\|_1 \\ &\text{s.t. } x'(l) - x'(l-1) \geq w(l-1); \quad 2 \leq l \leq N_r. \end{aligned} \tag{3.2}$$

The objective (3.2) describes the total movement of all cells along x-direction in a subrow after legalization. Its constraint of (3.2) satisfies the rule in legalization of no overlapping among cells. Moreover, the constraint preserves the order of cells, i.e., the relative order of cells in the legal placement is the same as the global placement.

Before legalization, two consecutive cells l and $l-1$ are overlapped; hence,

$$x(l) - x(l-1) \leq w(l-1). \tag{3.3}$$

Before solving the optimal problem (3.2)–(3.3), let's introduce a key theorem as follows. Before proving Theorem 1, let's introduce a simple lemma.

Lemma 1 *Given N_r ordered cells being overlapped in a subrow with the ordering preserved constraint, after legalization, if the J-th cell is the first cell whose left-side x-coordinate needs to be shifted to the right direction of its original position, the left-side x-coordinate of each cell l with $l > J$ must also be shifted to the right side of its original position.*

Proof: Let $x(l)$ and $x'(l)$ be the left-side x-coordinates of each cell l in the global placement and in the legalization result, respectively. Here, we need to prove $x'(l) \geq x(l)$ for each $l > J$.

For cell $J+1$, in the global placement, we have

$$x(J+1) - x(J) \leq w(J), \tag{3.4}$$

After legalization, we have

$$x'(J+1) - x'(J) \geq w(J). \tag{3.5}$$

Subtracting (3.4) from (3.5), we have

$$x'(J+1) - x(J+1) \geq x'(J) - x(J). \tag{3.6}$$

Since $x'(J) \geq x(J)$, (3.6) indicates $x'(J+1) \geq x(J+1)$. By the similar way, we can conclude that $x'(l) \geq x(l)$ for each $l > J$. $\square$

Theorem 1 *Given N_r ordered cells being overlapped in a subrow with the ordering preserved constraint, the optimal solution of the optimal problem (3.2)–(3.3) must satisfy*

$$x'(l) - x'(l-1) = w(l-1) \quad l = 2, \cdots, N_r$$

Here, $x(l)$ and $x'(l)$ are the lower-left corner x-coordinates of each cell l in the global placement and in the legalization result, respectively.

Proof: Given the J-th cell being the first cell which satisfies $x'(J) \geq x(J)$ and according to *Lemma* 1, (3.2) can be rewritten as

$$\begin{aligned}\sum_{l=1}^{N_r} |x'(l) - x(l)| &= \sum_{l=1}^{J-1} (x(l) - x'(l)) \\ &+ \sum_{l=J}^{N_r} (x'(l) - x(l)) \,. \end{aligned} \tag{3.7}$$

- $\sum_{l=1}^{J-1} (x(l) - x'(l))$: $x'(l) \leq x(l), 1 \leq l \leq J-1$

  After legalization, each $x'(l+1)$ can be rewritten as $x'(l+1) = x'(l) + w(l) + d_l$ for $1 \leq l \leq J-1$. The d_l is the distance between the right-side *x*-coordinate of cell l and the the left-side *x*-coordinate of cell $l+1$. Hence,

$$\begin{cases} x'(J-1) &= x'(J-2) + w(J-2) + d_{J-2}, \\ x'(J-2) &= x'(J-3) + w(J-3) + d_{J-3}, \\ & \vdots \\ x'(l+1) &= x'(l) + w(l) + d_l \end{cases}$$

  Summing up the above equations, $x'(l)$ can be represented as

$$x'(l) = x'(J-1) - \sum_{j=l}^{J-2} w(j) - \sum_{j=l}^{J-2} d_j \tag{3.8}$$

  To minimize $\sum_{l=1}^{J-1} (x(l) - x'(l))$ is equivalent to maximize each $x'(l)$ since $x'(l) \leq x(l)$. Furthermore, according to (3.8), to maximize $x'(l)$ is equivalent to simultaneously maximize $x'(J-1)$ and minimize $\sum_{j=l}^{J-2} d_j$ for each l.

  Taking $d_j = 0$ for $1 \leq j \leq J-2$, each $\sum_{j=l}^{J-2} d_j$ can be minimized to be zero which leads to

$$x'(l) - x'(l-1) = w(l-1), \quad l = 2, \cdots, J-1. \tag{3.9}$$

Therefore, we can conclude that to minimize $\sum_{l=1}^{J-1}(x(l) - x'(l))$ is equivalent to maximize $x'(J-1)$ under (3.9).

- $\sum_{l=J}^{N_r}(x'(l) - x(l))$: $x'(l) \geq x(l)$, $J \leq l \leq N_r$

After legalization, each $x'(l)$ can be rewritten as $x'(l) = x'(l-1) + w(l-1) + d_{l-1}$ for $J \leq l \leq N_r$. The d_{l-1} is the distance between the right-side *x*-coordinate of cell $l-1$ and the the left-side *x*-coordinate of cell l. Hence,

$$\begin{cases} x'(J+1) &= x'(J) + w(J) + d_J, \\ x'(J+2) &= x'(J+1) + w(J+1) + d_{J+1}, \\ &\vdots \\ x'(l) &= x'(l-1) + w(l-1) + d_{l-1} \end{cases}$$

Summing up the above equations, $x'(l)$ can be represented as

$$x'(l) = x'(J) + \sum_{j=J}^{l-1} w(j) + \sum_{j=J}^{l-1} d_j \tag{3.10}$$

To minimize $\sum_{l=J}^{N_r}(x'(l) - x(l))$ is equivalent to minimize each $x'(l)$ since $x'(l) \geq x(l)$. Furthermore, according to (3.10), to minimize $x'(l)$ is equivalent to simultaneously minimize $x'(J)$ and minimize $\sum_{j=J}^{l-1} d_j$ for each l.

Taking $d_j = 0$ for $J \leq j \leq N_r - 1$, each $\sum_{j=J}^{l-1} d_j$ can be minimized to be zero which leads to

$$x'(l) - x'(l-1) = w(l-1), \quad l = J+1, \cdots, N_r. \tag{3.11}$$

Therefore, we can conclude that to minimize $\sum_{l=J}^{N_r}(x'(l) - x(l))$ is equivalent to minimize $x'(J)$ under (3.11).

- Hence, to minimize $\sum_{l=1}^{N_r}|x'(l) - x(l)|$ is equivalent to simultaneously maximize $x'(J-1)$ and minimize $x'(J)$ under (3.9) and (3.11). Since, after legalization, $x'(J) - x'(J-1) \geq w(J-1)$, it can be easily to conclude that to minimize $\sum_{l=1}^{N_r}|x'(l) - x(l)|$, $x'(J)$ and $x'(J-1)$ must satisfy

$$x'(J) - x'(J-1) = w(J-1). \tag{3.12}$$

Finally, from (3.9), (3.11), and (3.12), we have

$$x'(l) - x'(l-1) = w(l-1), \quad l = 2, \cdots, N_r. \qquad \square \tag{3.13}$$

According to Theorem 1, the constraint in (3.2) can be simplified to

$$x'(l) - x'(l-1) = w(l-1); \quad l = 2, \cdots, N_r. \tag{3.14}$$

After several manipulations, (3.14) can be rewritten as

$$x'(l) = x'(1) + \sum_{j=1}^{l-1} w(j); \quad l = 2, \cdots, N_r. \tag{3.15}$$

Substituting (3.15) into the objective of (3.2), the Sum of Absolute Distances Solver (SADS) is constructed as follows.

$$\min \left\| \mathbf{x}'_{N_r} - \mathbf{x}_{N_r} \right\|_1 = \min \left(|r_1| + \cdots + |r_l| + \cdots + |r_{N_r}| \right). \tag{3.16}$$

Here, $|r_l| = |x'(1) + \sum_{j=1}^{l-1} w(j) - x(l)|$ which is the movement of cell l. After solving the minimization problem of (3.16), the optimal position of cell 1 can be obtained, and the optimal position of each cell l in the cluster can be obtained by (3.15), described in Subsection 3.5.2. Note that the cells are abutting in the cluster and there is no free space among them in the legal placement.

In general, the SADS can be cast as an linear programming problem after transformation; nevertheless, it takes much time to solve a large number of constraints of linear programming. Here, a fast approach for the problem is using the median concept to solve it. Firstly, $|r_l|$ can be rewritten as follows.

$$|r_l| = |x'(1) - \Delta x_l|, \tag{3.17}$$

where $\Delta x_l = x(l) - \sum_{k=1}^{l-1} w(k)$ for each cell l.

After that, the sequence $\{\Delta x_l\}_{l=1}^{N_r}$ is sorted to $\{\Delta x'_l\}_{l=1}^{N_r}$ by insertion sort or merging sort that depends upon the situation of one cell inserted in a cluster or two sorted clusters merged, respectively. The median of this sequence is derived as follows.

$$Q_{\frac{1}{2}}\left(\{\Delta x'_l\}_{l=1}^{N_r}\right) = \begin{cases} \Delta x'_{(N_r+1)/2}, & \textit{if } N_r \textit{ is odd;} \\ \frac{1}{2}(\Delta x'_{N_r/2} + \Delta x'_{N_r/2+1}), & \textit{if } N_r \textit{ is even.} \end{cases} \tag{3.18}$$

Choose $Q_{\frac{1}{2}}(\{\Delta x'_l\}_{l=1}^{N_r})$ as the x-coordinate position of the cell 1; therefore, the optimal position is obtained rapidly.

3.7 Runtime Complexity Analysis of *FastLegalize* Approach

Given a chip with global placement result constructed by NTUPlace3, firstly, the chip is divided into g_1 bins in $O(g_1)$ time, and Bin Merged Procedure algorithm presented in Figure 3.2 choose an adaptable bin-merged structure at most $O(g_1)$ time then. After that, it enters the Legalization Framework procedure presented in Figure 3.5. Note that g_1 is variable and k_i is constant for $i = 1, 2, 3, \cdots, i$ in the section.

Assume there are N cells in a chip and $k_1\frac{N}{g_1}$ cells in the bin structure. First, the Legalization Framework sorts the cells in bin structure in $O((k_1\frac{N}{g_1})\log(k_1\frac{N}{g_1}))$ time. Moreover, Legalization Framework takes the number of $k_1\frac{N}{g_1}$ times to do LegCore Function, presented in Figure 3.7. Assume LegCore takes O(M), the complexity for each bin structure is.

$$O((k_1\frac{N}{g_1})\log(k_1\frac{N}{g_1}) + k_1\frac{N}{g_1}M) \tag{3.19}$$

In the first step of LegCore function, the average number of Collapse called is $\frac{1}{2}N_r$ times. In the Collapse step, firstly, the $\frac{1}{2}N_r$ cells are sorted in $O(N_r)$ time by insertion sort or merging sort and the median of this cluster is picked up. After a series of steps in k_2 times, Collapse is called again since the cluster is overlapped with predecessor cluster; hence, the runtime needs $O((N_r)^2)$ in Collapse step. In the Optimal Position of Cells and Cost Function step, the worst case is $O(N_r^2)$ which is the same as the Collapse step; therefore, the analysis of complexity in the two steps is neglected. Above description summarized, the complexity in LegCore is.

$$O((N_r)^2) \tag{3.20}$$

Substituting the complexity of LegCore (3.20) into the complexity of Bin Merged Procedure (3.19), the complexity of overall algorithm is.

$$O((k_1\frac{N}{g_1})\log(k_1\frac{N}{g_1}) + k_1\frac{N}{g_1}(N_r)^2) \tag{3.21}$$

In our experimental observation, the time of first term of (3.21) is much smaller than the time of second term; therefore, the second term dominates the whole complexity. The complexity can be simplified to.

$$O(\frac{N}{g_1}(N_r)^2) \tag{3.22}$$

We take the bad situation into consideration; the complexity can be simplified to $O(N^2/g_1)$. Obviously, the *FastLegalize* is much faster than Abacus whose complexity is $O(N^2)$. In addition, the experimental results of Table 4.2 shows that the worst-case complexity of $O(N^2/g_1)$ is not reached.

Chapter 4

Experimental Results

The *FastLegalize* approach has been implemented in C++ programming language on a Intel Xeon 5160 3GHz workstation with 32GB memory. We compared our legalization approach with Abacus, which achieved the best published results compared with well-known legalization approach Tetris. Abacus, Tetris, and our *FastLegalize* all have the same purposes. One is to align all standard cells to subrows. Another is to legalize the standard cells with no overlapping. The other is that the relative order of the cells in each subrow is the same in the global placement. In the setting of specific threshold, it is set to 0.97 rather than 1 because some cells are much larger such that it can not be placed into the subrows. To avoid the situation, the value we set is lower than 1. Moreover, the section presents experimental results of Abacus and *FastLegalize* in legalizing NTUPlace3 [4] global placements of the 2005 International Symposium on Physical Design (ISPD'05) placement contest benchmark suites [1], and the NTUPlace3 is based on the log-sum-exp model.

One test case "bigblue3" in the benchmark has some movable cells which are not standard cells; therefore, the test case is not tested. The results of our legalization approach *FastLegalize* and the state-of-the-art method, Abacus, are shown in Table 4.2, and the benchmarks information is shown in the Table 4.1. In our experimental experience, the setting of the height and width of one bin is shown as follows.

$$h_b = h_r \sqrt{N_{row}} \tag{4.1}$$

$$w_b = 0.1 w_c \tag{4.2}$$

where h_r (h_b) is the height of one row (bin), N_{row} is the total number of rows of different height position in chip, and w_c (w_b) is the width of the chip (bin). In the table 4.1, "Chip

Width" and "Chip Height" are the width and height of the chip, respectively, "#Total Objects" is the total number of objects which consists of movable standard cells and fixed macros in the chip, "#Mov. Objects" is the total number of movable standard cells, "Total Mov." reports the summation movement of total movable cells between global placement and legalization result, "Average Mov." reports the movement of the cells between global placement and legalization, normalized to the average cell dimension of each circuit, "Max. Mov." reports the maximum movement of total movable standard cells between global placement and legalization, and "Time(s)" reports the runtime of algorithm of Abacus and *FastLegalize* individually. For more comparison, the experimental results of double times and four times the height of one bin are shown in Table 4.3. Although some cases of experimental results are better than before, it still proves the average experimental result of Table 4.2 are better.

Figure 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, and 4.7 show the global placement and legalization by *FastLegalize* of circuit adaptec1, adaptec2, adaptec3, adaptec4, bigblue1, bigblue2, and bigblue4, respectively. Table 4.2 demonstrates that our approach using Bin Merged Procedure, Legalization Framework, LegCore, and SADS can legalize the circuits fast and effectively. Compared with the state-of-the-art method, Abacus, *FastLegalize* can reduce the total and average movement of cells to be 48% , and save the largest movement of cells to be 140% in average. It demonstrates the structure of CSBMS/SSBMS can effectively limit the movable scope of cells. Moreover, *FastLegalize* can obtain average 1.11× runtime speed up. It also demonstrates the structure of CSBMS/SSBMS can reduce the complexity effectively, and SADS does not take much time but is as exact as expected. Hence, our *FastLegalize* can minimize the disturbance and legalize the circuits fast and effectively.

	Circuit	Chip Width	Chip Height	#Total Objects	#Mov. Objects
Benchmarks [1]	adaptec1	10692	10680	211447	210904
	adaptec2	14054	14040	255023	254457
	adaptec3	23190	23328	451650	450927
	adaptec4	23190	23328	496045	494716
	bigblue1	10692	10680	278164	277604
	bigblue2	18690	18792	557866	534782
	bigblue4	32190	32328	2177353	2169183

Table 4.1: 2005 ISPD Placement Contest and Benchmark Information [1]

Benchmarks [1]	*FastLegalize*				Abacus			
Circuit	Total Mov.	Average Mov.	Max. Mov.	Time (s)	Total Mov.	Average Mov.	Max. Mov.	Time (s)
adaptec1	11074785.9	0.29	2810.3	10.4	16802629.8	0.45	8645.3	28.6
adaptec2	28863882.1	0.85	7338.2	11.1	28060678.4	0.83	13446.6	33.4
adaptec3	181479037.4	2.60	11008.2	92.4	355133608.3	5.08	21253.3	42.9
adaptec4	55234207.3	0.73	6918.3	45.5	120071050.5	1.59	21806.6	79.8
bigblue1	18703773.3	0.44	5239.1	20.8	16964490.2	0.40	7160.3	70.1
bigblue2	34552315.9	0.42	3712.4	43.6	56279868.5	0.68	12741.1	93.7
bigblue4	866742682.6	3.01	14500.7	789.36	998137528.9	3.47	28891.3	1019.5
Comparison	**1**	**1**	**1**	**1**	**1.48**	**1.48**	**2.40**	**2.11**

Table 4.2: Comparison for the Proposed *FastLegalize* Procedure with Abacus [2]

Benchmarks [1]	*FastLegalize* $(2 \times h_b)$				*FastLegalize* $(4 \times h_b)$			
Circuit	Total Mov.	Average Mov.	Max. Mov.	Time (s)	Total Mov.	Average Mov.	Max. Mov.	Time (s)
adaptec1	23992803.7	0.64	4715.2	9.9	22721983.0	0.60	5027.7	12.0
adaptec2	26039069.4	0.77	7880.3	15.3	41338824.1	1.22	9139.6	22.1
adaptec3	175614423.4	2.51	10368.9	114.6	164744365.5	2.36	9579.1	116.7
adaptec4	78826003.5	1.04	6845.5	46.1	71919265.3	0.95	7821.9	75.6
bigblue1	18905391.3	0.44	4320.2	34.8	23326904.5	0.55	6250.8	37.4
bigblue2	47594610.9	0.58	7725.1	39.8	39996324.4	0.48	6881.6	35.6
bigblue4	803083074.5	2.79	10999.1	2462.65	870988654.2	3.02	9738.9	1509.1
Comparison	**1.25**	**1.25**	**1.19**	**1.45**	**1.30**	**1.30**	**1.25**	**1.49**

Table 4.3: Experimental Results of *FastLegalize* $(2 \times h_b)$ and *FastLegalize* $(4 \times h_b)$

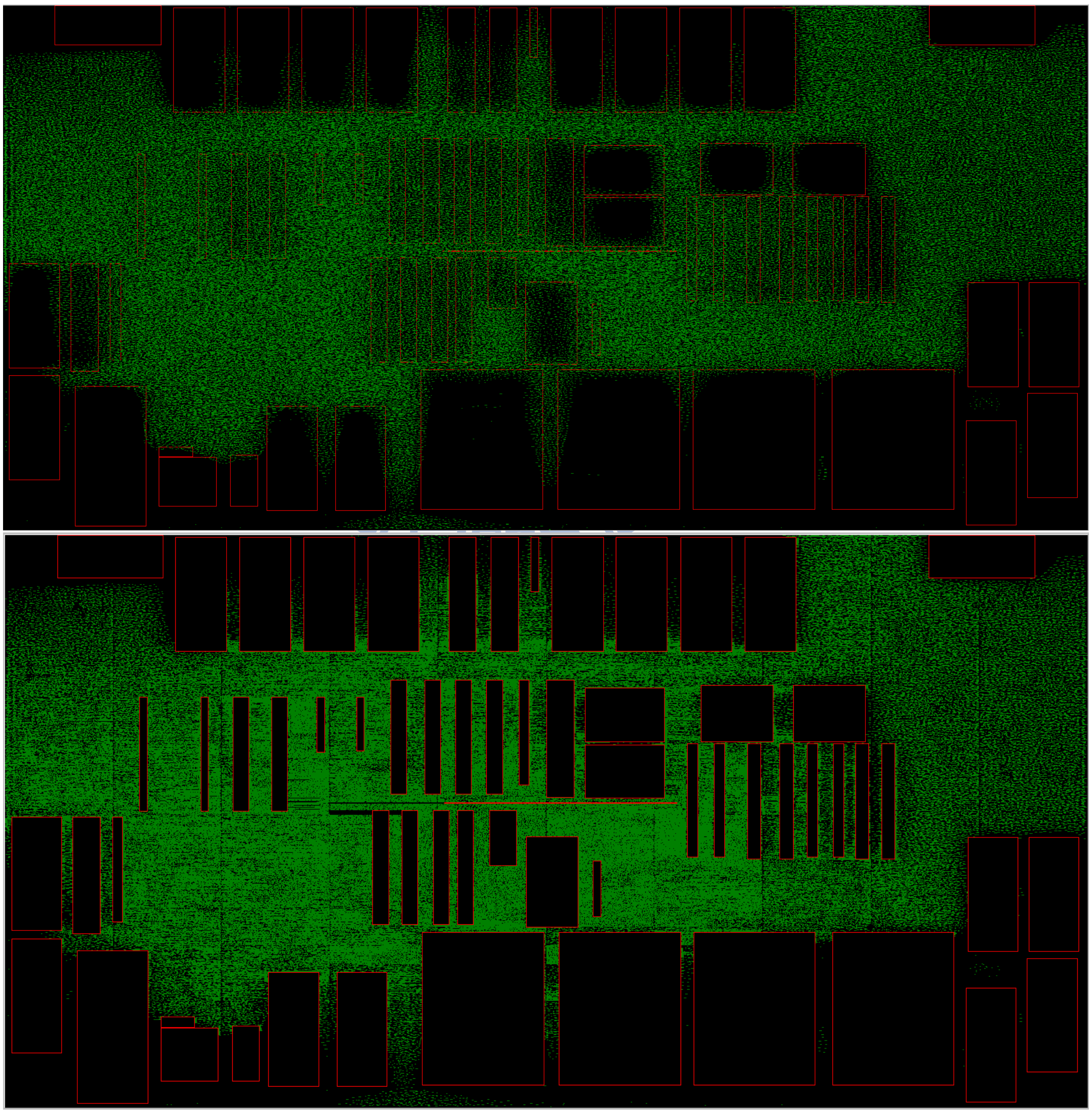

Figure 4.1: (a) Global placement result of circuit adaptec1 (b) Legalization result of circuit adaptec1

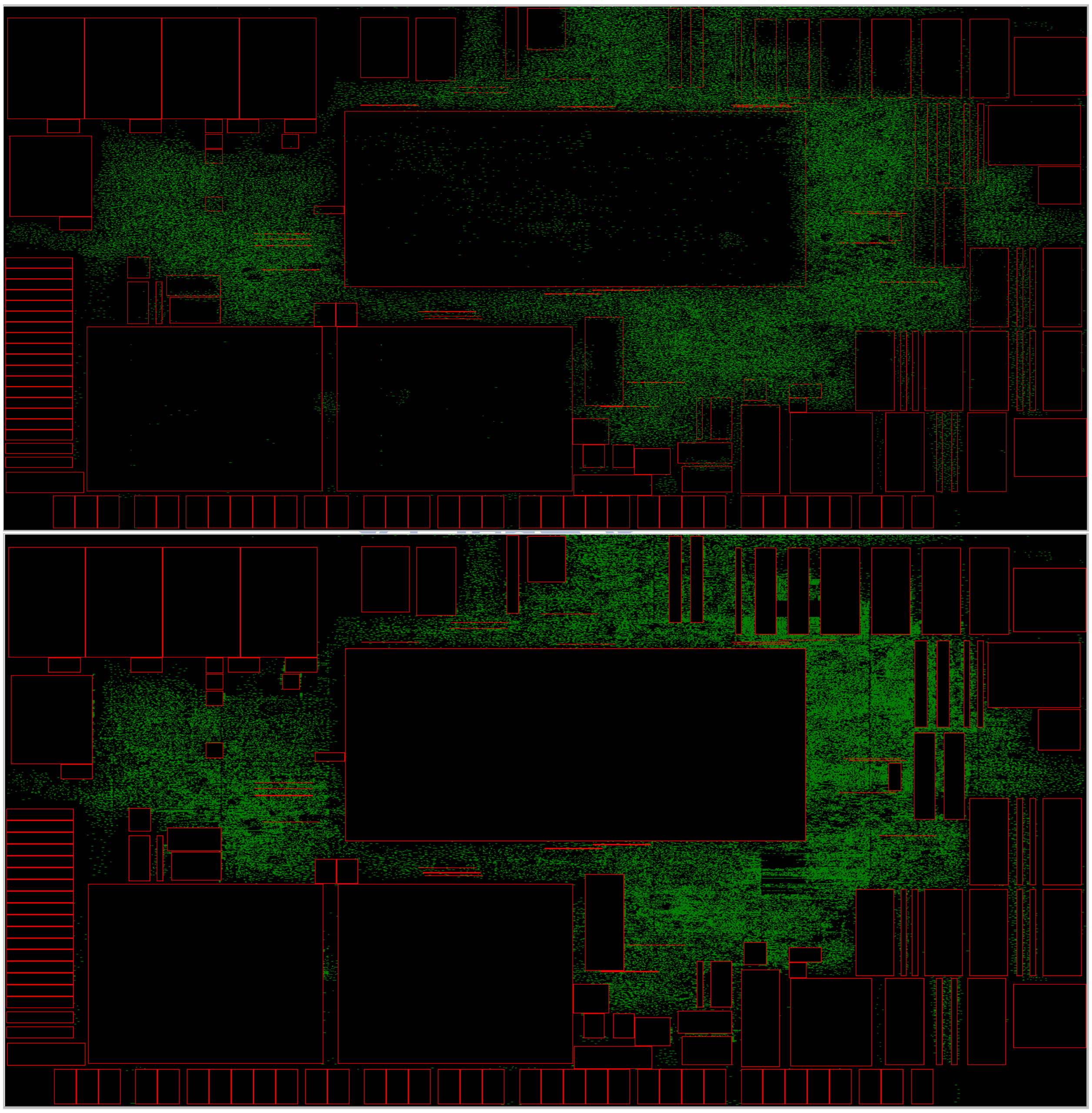
Figure 4.2: (a) Global placement result of circuit adaptec2 (b) Legalization result of circuit adaptec2

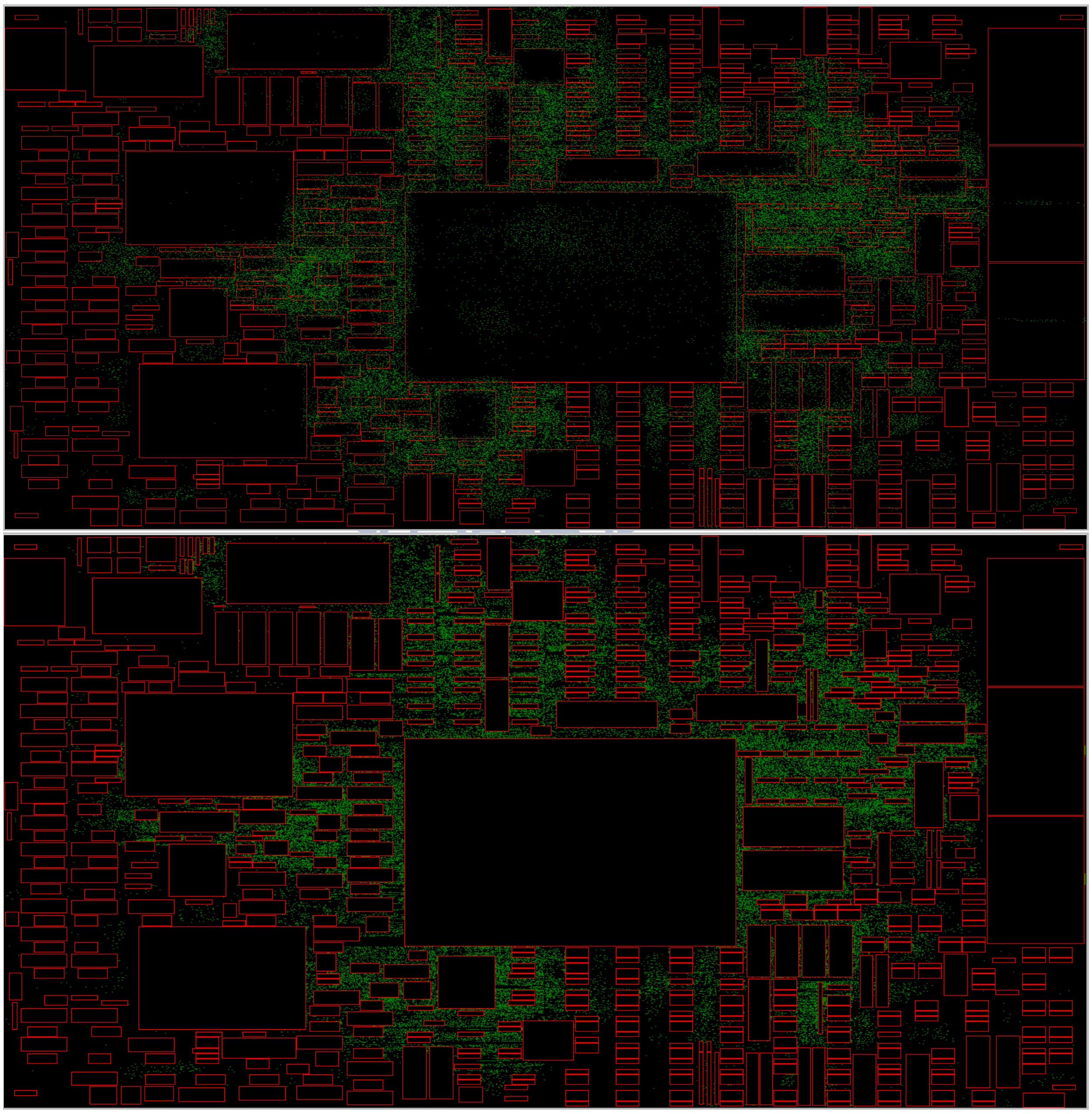
Figure 4.3: (a) Global placement result of circuit adaptec3 (b) Legalization result of circuit adaptec3

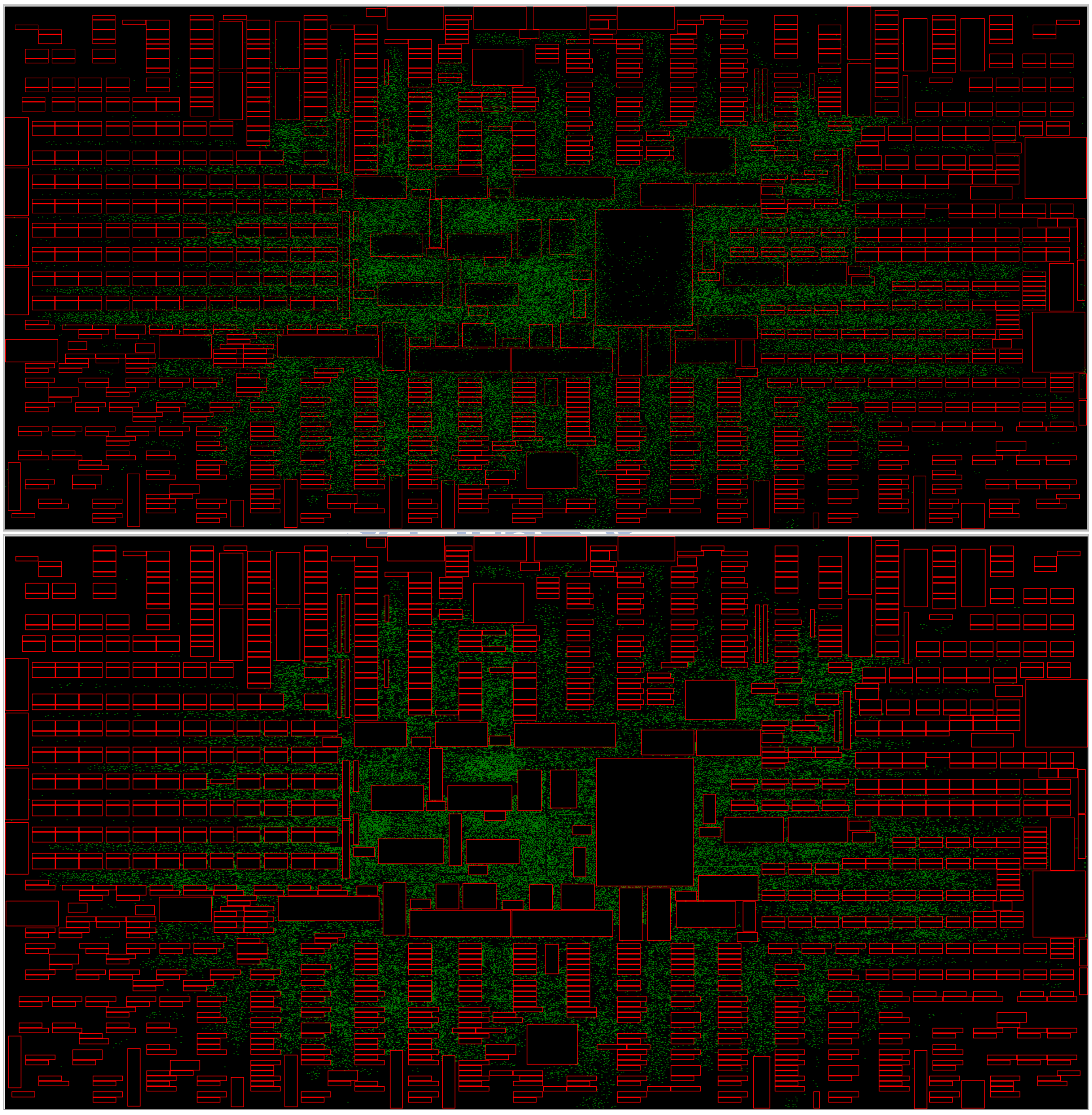

Figure 4.4: (a) Global placement result of circuit adaptec4 (b) Legalization result of circuit adaptec4

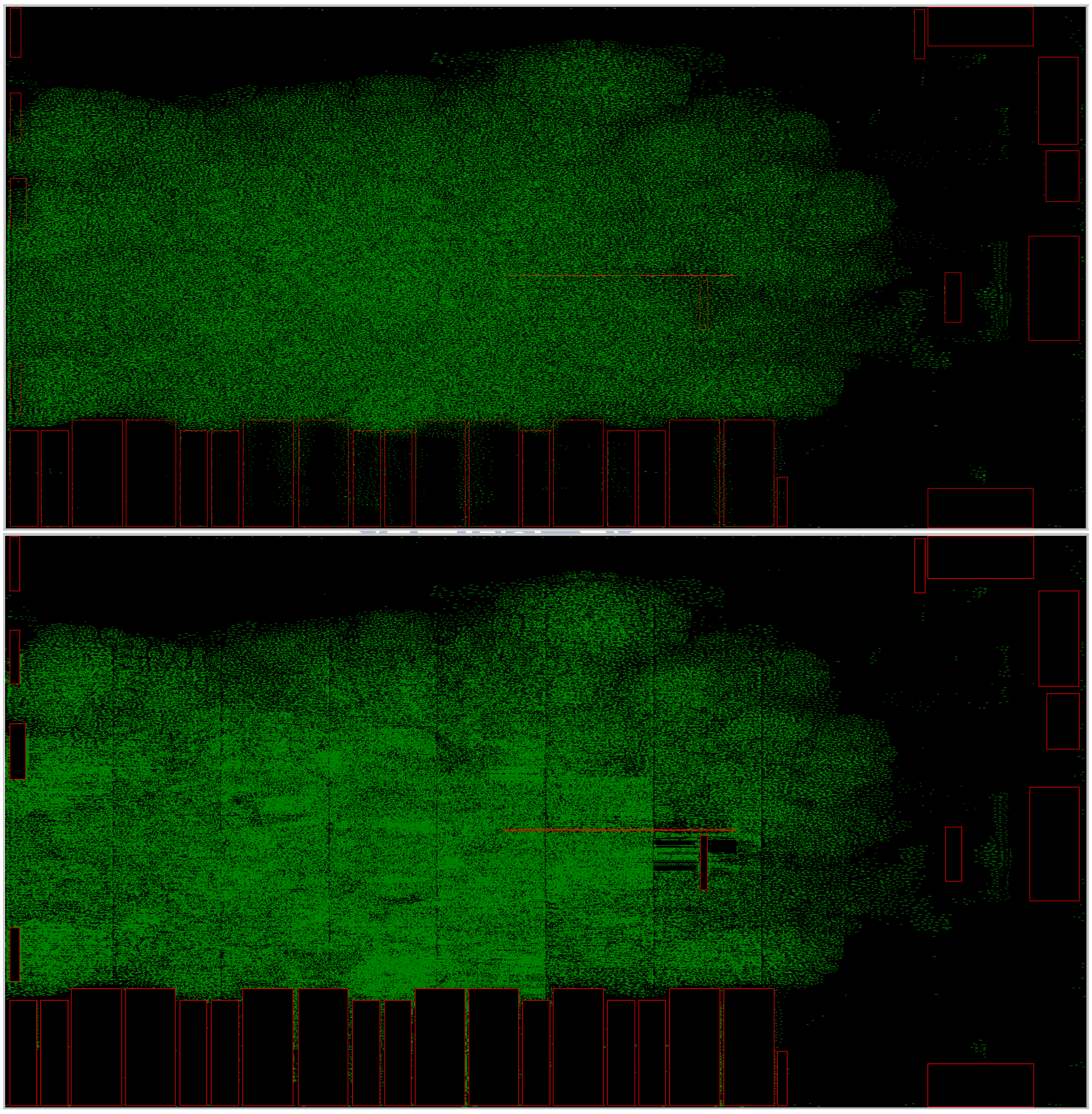

Figure 4.5: (a) Global placement result of circuit bigblue1 (b) Legalization result of circuit bigblue1

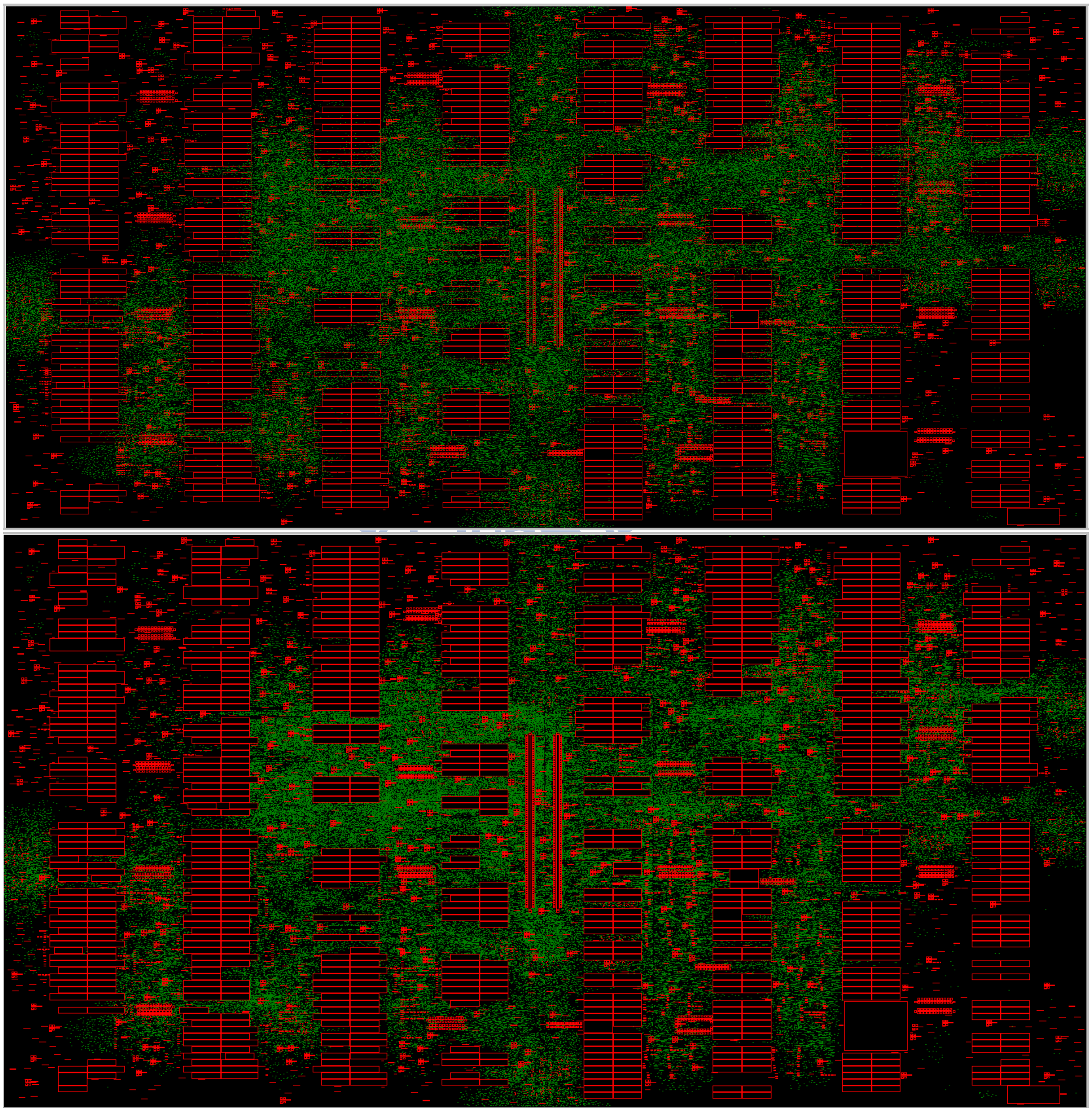
Figure 4.6: (a) Global placement result of circuit bigblue2 (b) Legalization result of circuit bigblue2

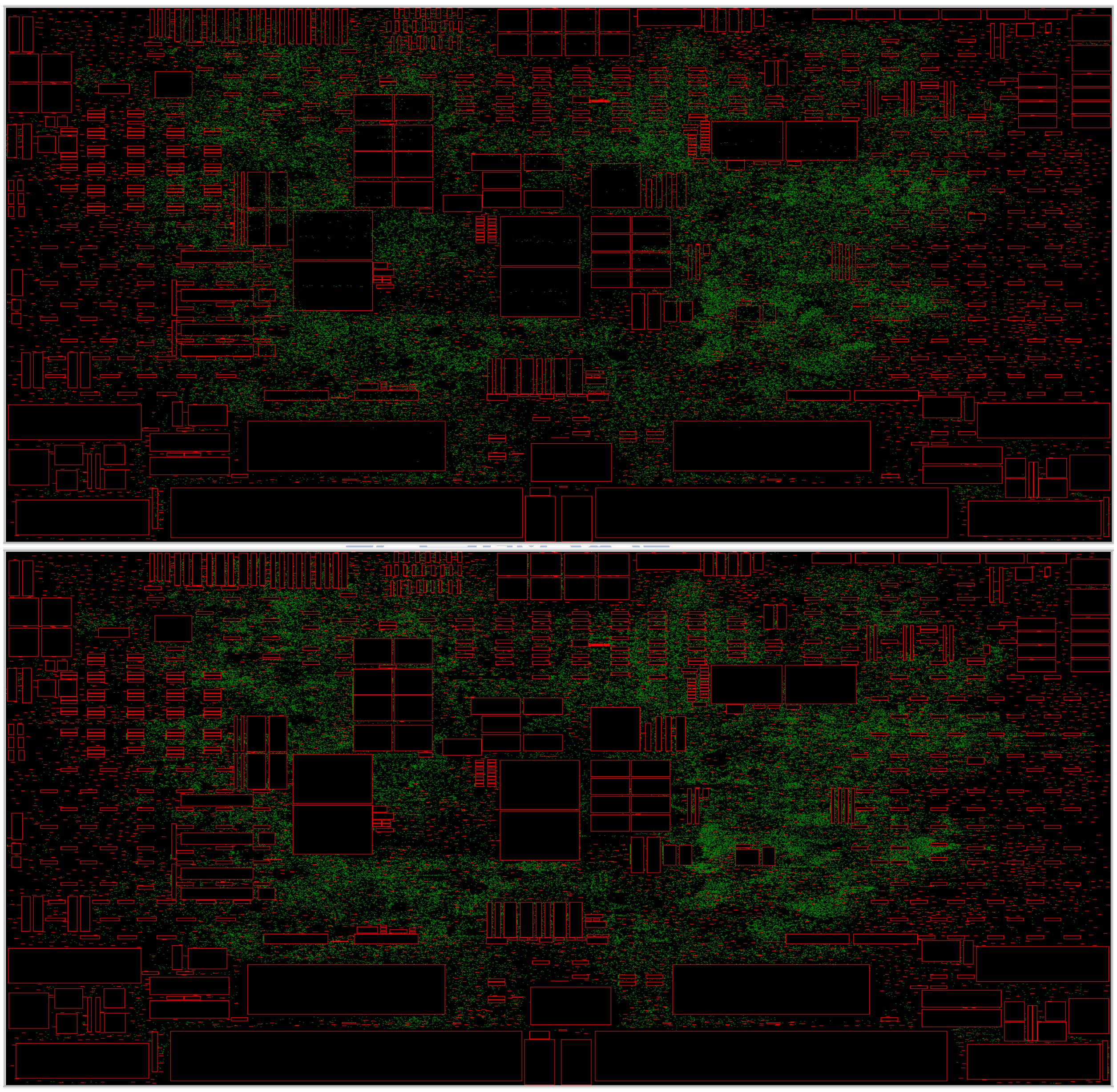

Figure 4.7: (a) Global placement result of circuit bigblue4 (b) Legalization result of circuit bigblue4

Chapter 5

Conclusion

In the thesis, we have presented *FastLegalize*, which minimizes the total movement of cells for minimal disturbance in placement legalization stage, applicable for standard cell circuits. The quadratic programming model presented before is not as exact as expected. Therefore, the SADS is proposed to improve it. Moreover, CSBMS/SSBMS in Bin Merged Procedure is proposed to limit the movable scope of cells. All cells in each row are placed optimally by minimizing the total movement. It reduces the total movement of total cells about 48% in average compared with state-of-the-art method, Abacus. In the future work, the *FastLegalize* would legalize the standard cells with timing-driven consideration.

Bibliography

[1] International symposium on physical design 2005 placement contest. *http://www.sigda.org/ispd2005/contest.htm*, April 2005.

[2] Peter Spindler, Ulf Schlichtmann, and Frank M. Johannes. Abacus: Fast legalization of standard cell circuits with minimal movement. In *Proceedings ACM/SIGDA International Symposium on Physical Design*, pages 47–53, April 2008.

[3] Paul Villarrubia. Important placement considerations for modern vlsi chips. In *Proceedings ACM/SIGDA International Symposium on Physical Design*, pages 6–6, 2003.

[4] Tung-Chieh Chen, Zhe-Wei Jiang, Tien-Chang Hsu, Hsin-Chen Chen, and Yao-Wen Chang. Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(7):1228–1240, July 2008.

[5] Natarajan Viswanathan, Min Pan, and Chris Chu. Fastplace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control. In *Proceedings Asia and South Pacific Design Automation Conference*, pages 135–140, January 2007.

[6] Tao Luo and David Z. Pan. Dplace2.0: A stable and efficient analytical placement based on diffusion. In *Proceedings Asia and South Pacific Design Automation Conference*, pages 346–351, 2008.

[7] Dwight Hill. Method and system for high speed detailed placement of cells within an integrated circuit design. *U.S. Patent 6370673*, April 2002.

[8] Chen Li, Min Xie, Cheng-Kok Koh, Jason Cong, and Patrick H. Madden. Routability-driven placement and white space allocation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(5):858–871, May 2007.

[9] Ateen Khatkhate, Chen Li, Ameya R. Agnihotri, Mehmet C. Yildiz, Satoshi Ono, Cheng-Kok Koh, and Patrick H. Madden. Recursive bisection based mixed block placement. In *Proceedings ACM/SIGDA International Symposium on Physical Design*, pages 84–89, 2004.

[10] Andrew B. Kahng and Qinke Wang. Implementation and extensibility of an analytic placer. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(5):734–747, May 2005.

[11] Chen Li and Cheng-Kok Koh. On improving recursive bipartitioning-based placement. *Technical Report TR-ECE 03-14, Purdue University*, December 2003.

[12] Jason Cong and Min Xie. A robust detailed placement for mixed-size ic design. In *Proceedings Asia and South Pacific Design Automation Conference*, pages 188–194, 2006.

[13] Haoxing Ren, David Z. Pan, Charles J. Alpert, Paul G. Villarrubia, and Gi-Joon Nam. Diffusion-based placement migration with application on legalization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(12):2158–2172, December 2007.

[14] Tao Luo, Haoxing Ren, Charles J. Alpert, and David Z. Pan. Computational geometry based placement migration. In *Proceedings ACM/IEEE Design Automation Conference*, pages 41–47, 2007.

[15] Carl Sechen and Alberto Sangiovanni-Vincentelli. Timberwolf 3.2: A new standard cell placement and global routing package. In *Proceedings ACM/IEEE Design Automation Conference*, pages 432–439, 1986.

[16] Jason Cong and Min Xie. Dragon2000: Standard-cell placement tool for large industry circuits. In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 260–263, 2000.

[17] M. Sarrafzadeh and M. Wang. Nrg: Global and detailed placement. In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 532–537, 1997.

[18] Min Pan, Natarajan Viswanathan, and Chris Chu. An efficient and effective detailed placement algorithm. In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 48–55, 2005.

[19] Ulrich Brenner, Anna Pauli, and Jens Vygen. Almost optimum placement legalization by minimum cost flow and dynamic programming. In *Proceedings ACM/SIGDA International Symposium on Physical design*, pages 2–9, 2004.

[20] Ulrich Brenner and Jens Vygen. Legalizing a placement with minimum total movement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(12):1597–1613, December 2004.

[21] Andrew B. Kahng, Paul Tucker, and Alex Zelikovsky. Optimization of linear placements for wirelength minimization with free sites. In *Proceedings Asia and South Pacific Design Automation Conference*, pages 241–244, 1999.

[22] Ulrich Brenner and Jens Vygen. Faster optimal single-row placement with fixed ordering. In *Proceedings Design Automation and Test in Europe*, pages 117–121, 2000.

[23] Ameya Agnihotri, Mehmet Can Yildiz, Ateen Khatkhate, Ajita Mathur, Satoshi Ono, and Patrick H. Madden. Fractional cut: Improved recursive bisection placement. In *Proceedings International Conference on Computer Aided Design*, pages 307–310, November 2003.

[24] Andrew B. Kahng, Igor L. Markov, and Sherief Reda. On legalization of row-based placements. In *Great Lakes Symposium on VLSI*, pages 214–219, 2004.

[25] Sung-Woo Hur and John Lillis. Mongrel: Hybrid techniques for standard cell placement. In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 165–170, 2000.

[26] Konrad Doll, Frank M. Johannes, and Kurt J. Antreich. Iterative placement improvement by network flow methods. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(10):1189–1200, October 1994.