THE HONG KONG POLYTECHNIC UNIVERSITY
DEPARTMENT OF ELECTRONIC AND INFORMATION ENGINEERING

Fundamental Research on Electronic Design Automation in VLSI Design - Routability

Lu Jingwei

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Philosophy

February, 2010

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

_____ (Name of student)

Abstract

As the feature size of integrated circuits is revolutionized into nanometer scale, delay of interconnection has become the dominant factor instead of the transistor internal delay. As a result, new demands on interconnection have been proposed to the developers, and they usually enhance the routability of the chip so as to improve the performance of interconnection. Under the general research topic of routability, our work is focused on congestion prediction, clock network synthesis, clock gating design and global routing. They are all critical steps regarding routability concerns in the VLSI physical design.

In the early stages of the physical design, congestion prediction is necessary for the routability evaluation. An accurate estimation of a placement result is an effective metric to evaluate the behavior of the corresponding placer. In our work we propose three models, shortest Manhattan distance (SMD) model, Detour model and 3-step approach, for congestion prediction. The two major techniques applied in modern global routers, the detoured routing as well as the rip-up and rerouting, are considered in our work for further enhancement on estimation accuracy. The experimental results of our work present a progress on the performance compared to the previous congestion models.

The behavior (timing delay) of clock network synthesis (CNS) is mainly determined by the clock skew and the PVT (Process, Voltage and Temperature) variation factors. In our work, we develop two new clock network synthesizers (DMST and DMSTSS) with several novel techniques to tackle these issues.

A dual-MST based perfect matching and a hierarchical buffer sizing are proposed to handle the clock latency range (CLR), which is the major metric for performance evaluation. An iterative buffer insertion approach and a dual-MZ blockage handling technique are developed for a proper distribution of buffers and wires. Internal nodes of the clock tree are relocated based on the delay estimation by SPICE simulation. The clock skew can be further reduced in this procedure. Slew table construction is designed to conform to the constraint on slew rate. In the experimental results it is shown that our synthesizers can effectively reduce the CLR in a much shorter runtime.

In the modern synchronous digital circuits, the clock network consumes a great share of the total power cost. Therefore, it is necessary to engage masking gates to reduce its power usage. This technique turns off the according clock tree sections during their idle periods. In the previous clock gating works, switched capacitance is a major metric to denote the power usage of the clock network and the according controller network. Two clock gating works, HKPUcg and HKPUst, are proposed in this thesis. HKPUcg aims at minimizing the switched capacitance, and the objective of HKPUst is reducing the clock skew. Two novel methods of power aware topology generation are proposed, respectively. Moreover, a new decision technique on gate insertion is developed to further reduce the switched capacitance and balance the delay difference. From the experimental results, we can see that our clock gating works can effectively reduce the total power usage. By SPICE simulation, the clock skew is small.

Among the modern global routers, the technique of iterative rip-up and rerouting is widely applied. Based on this technique, we develop two methods of dynamic steiner point relocation and edge-based maze routing to further reduce the overflow and shorten the wirelength. The first approach is implemented in constant time with a new data structure constructed for pins, steiner points and subnet connection. The second approach is built up based on the

propagation among the global edges instead of global bins. From the experimental results, we can see that our router is efficient and robust compared to the previous state-of-the-art global routers.

Acknowledgements

At first, I would like to express the deepest gratitude to my chief supervisor, Dr. Bruce Chiu-Wing Sham, for his excellent direction, patient instruction and generous help throughout the past two years. During my MPhil study, Dr. Sham not only enriches my academic experiences, broadens my horizon of research, but also impresses me with his kindness and tolerance. Without him I could not finish my MPhil study, and it is a great honor of mine to study under his supervision.

Secondly, I would like to express the sincere thanks to my co-supervisor, Prof. Evangeline Fung-Yu Young, for her guidance and cares to me. Besides research instructions, Prof. Young also gives me encouragement to pursue higher goals in my career life. Without her I could not proceed my study at VLSI CAD, let alone my future works. It is beyond my words to express my appreciation to her.

Thirdly, I would like to thank my colleague, William Chow-Wing Kai, for his support to my research work. I have benefited a lot from the discussion with him regarding academic problems and living concerns. It is a memorable experience of mine to work with him.

Meanwhile, I appreciate the cares and helps from my parents during my study in Hong Kong. I could never pursue my oversea education without their comprehension and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the modern time, integrated circuit (chip) is widely applied in the electronic equipments. Almost every digital appliance, like computer, camera, music player or mobile phone, has one or several chips on its circuit board. VLSI, the acronym of very-large-scale integration, is the process of combining a huge amount of transistor-based circuits into a single chip. Complex of electronic components are designed, specified then fabricated on the substrate, which is made of pure semiconductor materials. Physical design is at the lowest level of the VLSI design flow, it is the process of determining the actual location of all the active devices and interconnecting the pins inside the boundary of a VLSI chip. As the improvement on the craft of semiconductor manufacture proceeds into deep-submicron and nanometer scale and the enlargement of integration scale into billions of transistors, designing demands of much stricter rules (power usage, process variation, timing closure) have been proposed. It is no longer applicable for the manual operation to deal with problems of such huge difficulty.

Instead of traditional manual design, electronic design automation (EDA) is applied to automate the design process of semiconductors and improve the efficiency of the work. The major objective of EDA is to develop a category of

CAD (computer aided design) software tools. Meanwhile, the according performance simulation can be applied in PC or supercomputers, which is very attainable. In the procedure of EDA, fundamental comprehension on the VLSI technology and the knowledge on the feature attributes are important skills for the designers. Therefore, they can efficiently fill the gap between system specification and chip manufacture. By means of problem formulation, algorithm devision and performance evaluation, designers are supposed to build up a program to automate the physical design in a computer.

Due to the prevalent application of nanometer-scale technology in the modern integrated circuits, the excessively high density of interconnections will result in negative effect towards the routability of the chip. This is because the over-congestion of wires will severely damage the quality of the signal transmission in each network, therefore impair the circuit performance. As a result, routability optimization has become a major concern in the VLSI physical design. In our research work, routability is improved in several correlated steps: congestion prediction, clock tree synthesis and global routing.

1.2 Our Contribution

In VLSI physical design, our research work is focused on the topic of routability. It includes both the estimation and the implementation of the interconnections of the networks. Our contribution is composed of four parts:

1. Congestion Prediction. We propose three models (SMD model, detour model and 3-step approach) to evaluate the congestion in early stages. Based on the packing results of a placer, the grade of congestion of all the tiles on the chip can be estimated by our prediction models. The result of prediction is an effective metric to evaluate the performance of the according placer, therefore enhance the routability of the chip and facilitate the network interconnection (routing) in later stages.

2. Clock Network Synthesis. In our work, we develop two synthesizers (DME-STSS and DME-ST). Overall six novel techniques are proposed to improve the performance. A merging node relocation approach is devised to reduce the clock skew, which is the major concern in CNS. By our blockage handling approach, distribution problems for buffer insertion are successfully solved, and the additional resource cost is very low. Besides, the capacitance usage, although a less important concern, is also reduced by our merging approach.

3. Clock Gating Design. It is an extended work based on the clock network synthesis. In clock gating design, the power usage of the network becomes the major concern. In this thesis, two clock gating works (HKPUcg and HKPUst) are proposed. An improved power aware topology of the clock tree is implemented. Meanwhile, a decision technique on clock gate insertion is developed to reduce the resultant switched capacitance thus cut down the power usage. Our synthesizers can guarantee exact zero skew without violation on the slew rate constraint.

4. Global Routing. We propose a new global router, HKPUgr, for inter-bin connection of all the networks in the chip. We developed two novel methods, a dynamic steiner point relocation technique as well as an edge-based maze routing technique, to reduce the wirelength and via length. In each iteration of iterative rip-up and rerouting, our approach is proved to achieve the optimum solution based on a common composite cost function of global routing.

1.3 Outline of Thesis

This thesis is organized as follows. We give the introduction of our thesis in chapter 1. The background knowledge regarding this thesis is presented in

chapter 2, with the overview of the VLSI physical design and some discussions on the related performance metrics. Previous research works are reviewed and analyzed in chapter 3. Three models of congestion prediction is discussed in chapter 4 with comprehensive analysis. Two clock network synthesizers with six technical enhancements are proposed in chapter 5. Clock gating design is proposed in chapter 6 to reduce the power usage of the clock network by gate insertion. A new global router is proposed in chapter 7 to reduce the total wirelength. Finally, we reach our conclusion in chapter 8.

Chapter 2

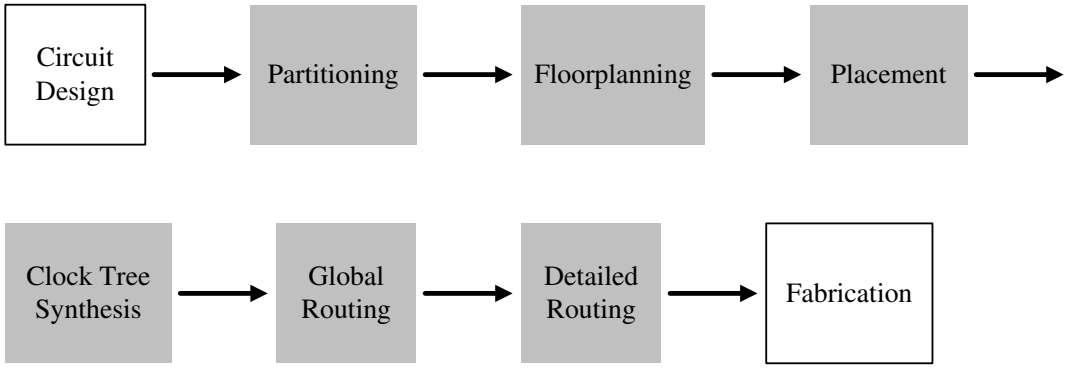
Background

2.1 Overview of VLSI Physical Design

Modern VLSI design is usually divided into four concatenated stages: behavioral design, structural design, logic design and physical design. In the VLSI physical design, the source code at register transfer level (RTL) in hardware description language (HDL) is converted into specific circuit layout for manufacture. VLSI physical design is typically solved in a hierarchical framework. For problem simplification with less concerns of optimization involved, each design stage would be mostly independently improved. Meanwhile, correlated optimization concerns among various stages should also be engaged, in order to maintain the total layout problem manageable for the subsequent stages (fabrication). The procedure of VLSI physical design is generally composed of five steps: partitioning, floorplanning, placement, clock tree synthesis, global routing and detailed routing. The design flow is shown by the gray rectangles in figure 2.1. Sometimes partitioning, floorplanning and placement are summarized into a superior stage of placement, and clock tree synthesis, global routing and detailed routing are summarized into a superior stage of routing.

Partitioning divides a circuit into smaller parts. The objective is to restrict each part within a prescribed range. Meanwhile, the amount of interconnections among different components is minimized so as to enhance the routability

Figure 2.1 VLSI physical design flow.



and reduce the power usage. Partitioning is a fundamental step of the VLSI physical design, it transforms a large problem into smaller ones of manageable sizes. In the literature, the research works on partitioning mainly include iterative partitioning algorithms [45, 28, 48, 81] and ratio cut approaches [73, 51].

Floorplanning is applied after partitioning to determine the approximate location of each module in the chip area. A good approach of floorplanning should minimize the total chip area, facilitate the following routing stage and reduce the signal delay. Modules with higher density connections should be placed closer to each other. In retrospect to the past, various approaches concerning floorplanning have been proposed. They are mainly composed of rectangular dual graph approach [47], planar triangulated graph approaches [90, 93], hierarchical approach [77], simulated annealing approach [97] and floorplan sizing approach [67, 87].

In placement, each module has fixed shape and terminal locations. The main target is to determine the optimal position of every module on the chip. The placement algorithms can be divided into two main categories: iterative improvement and constructive placement. General methods include force-directed [2, 34, 66], simulated annealing [82], partitioning [23] and resistive network [16].

The major target of clock network synthesis is to connect all the synchronous signal receivers together with the signal source. Buffer insertion, sizing and wire customization are also engaged to synthesize the work and further improve the timing performance. The basic two approaches of clock tree construction are the top-down procedure [40] and the bottom-up procedure [41]. There are also two general methods of signal delay computation: linear delay estimated by wirelength and Elmore RC delay model [26].

In global routing, the grid is decomposed into small rectangles, which are named global bins. Only inter-bin connection among the pins of each network is applied in global routing. It is aimed at reducing the degree of congestion and the total wirelength. A good global router will evenly distribute the congestion over the whole routing area. The general routing methods include multi-commodity flow-based approaches [6, 85] and single-turn routing approach [43].

Detailed routing follows the global routing and implements the interconnections inside the global bins. Track assignment between global bins is also completed. The general approaches include channel routing [78] and hierarchical routing [4].

2.2 Metrics Analysis

In this section, we introduce the major metrics for performance evaluation in this thesis. Generally, the evaluation is based on the result regarding two items: (1) timing delay (2) power consumption. Both of the two metrics should be minimized in order to improve the performance of our research work.

2.2.1 Signal Delay Models

In clock tree synthesis, there are plenty of delay models in different orders. By applying these models, signal delay at any point of the clock network can

Figure 2.2 A clock inverter and its corresponding RC delay model

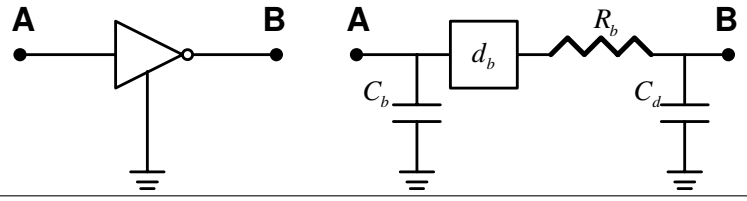
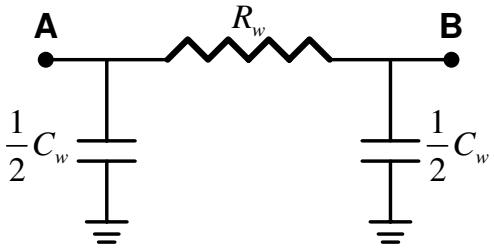


Figure 2.3 π-model of a single wire



be calculated. Linear delay model is a simple model for delay computation and balancing. The pathlength between a clock sink to the source is used to denote its according timing delay. Besides, Elmore RC model [26] is a more effective delay model. Compared to the linear model, Elmore RC model is more accurate together with increased complexity of computation.

An example of RC delay model of a buffer is shown in figure 2.2. The corresponding formula to calculate the buffer delay is shown as below

$$D_B = d_b + R_b \times C_d \tag{2.1}$$

where C_d means the load capacitance at point B, d_b is the internal delay of a buffer and R_b is the driver resistance of a buffer.

The delay model of a wire can be obtained in a similar way. We usually use π-model to denote the wire delay case, as illustrated in figure 2.3. Based on the Elmore delay model, the following equation is used to calculate the delay of a wire,

$$D_w = R_w \times \left(\frac{1}{2} \times C_w + C_d\right) \tag{2.2}$$

where C_d denotes the load capacitance at point B, R_w denotes the unit resistance of a wire and C_w denotes the unit capacitance of a wire.

2.2.2 Clock Skew and Clock Latency Range(CLR)

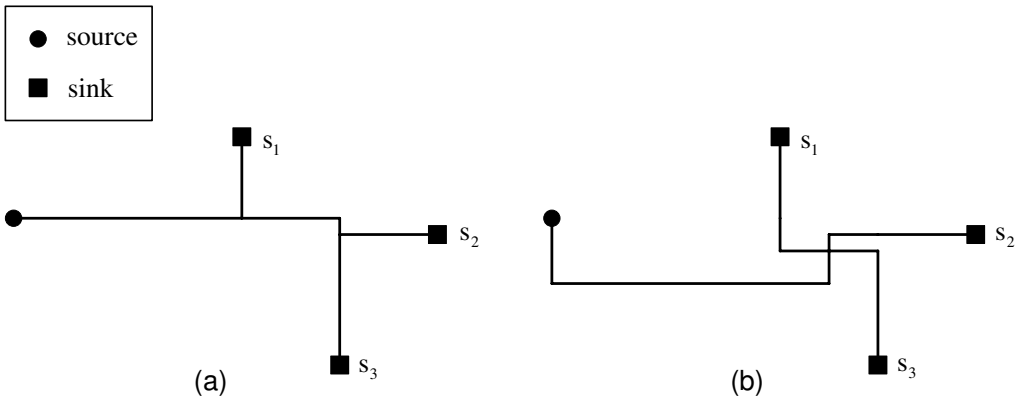
Let S denote the set of sinks, $S = \{s_1, \ldots, s_{|S|}\}$. s_i is the ith sink, and d_{s_i} represents the internal delay between the source and the sink s_i. The skew of a clock network means the maximum difference of the source-to-sink delay among all the sinks. To minimize the skew, we need to build a clock network with all d_{s_i} as close as possible. Notice that the skew is not determined by the average delay of the sinks, but by the difference between the maximum delay and the minimum delay. The equation for clock skew (C_s) calculation is shown as below.

$$C_s = \max\{d_{s_i} | \forall s_i \in S\} - \min\{d_{s_i} | \forall s_i \in S\} \tag{2.3}$$

Two examples in linear delay model are shown in figure 2.4. Figure 2.4(a) is a non-equidistant solution. We can find that $d_{s_1} < d_{s_2} < d_{s_3}$, and the pathlength skew is $(d_{s_3} - d_{s_1})$. However, in figure 2.4(b) $d_{s_1} = d_{s_2} = d_{s_3}$ and the pathlength skew is zero. Therefore, the equidistant one is better than the non-equidistant one, although its total wirelength is longer.

Due to the PVT (process, voltage and temperature) variation, the actual performance of the clock network would be worse than the theoretical estimation. In ISPD 2009 Clock Network Synthesis Contest [88, 39], a new metric named clock latency range (CLR) is formulated, concerning voltage variation additionally. Assume that the supplied voltage at each buffer vary between V_{dd_1} and V_{dd_2}. Therefore, two voltage sources V_{dd_1} and V_{dd_2} are applied in two simulation tests independently to simulate the voltage uncertainty. The metric of CLR is thus determined by the difference between the maximal and minimal clock skew values under these two given voltage supplies. CLR is the major

Figure 2.4 (a) Non-equidistant clock tree (b) Equidistant clock tree.



metric for performance evaluation in our CNS work.

$$p_{1_{max}} = \max\{d_{s_i} | \forall s_i \in S, V_{dd_1}\} \tag{2.4a}$$

$$p_{1_{min}} = \min\{d_{s_i} | \forall s_i \in S, V_{dd_1}\} \tag{2.4b}$$

$$p_{2_{max}} = \max\{d_{s_i} | \forall s_i \in S, V_{dd_2}\} \tag{2.4c}$$

$$p_{2_{min}} = \min\{d_{s_i} | \forall s_i \in S, V_{dd_2}\} \tag{2.4d}$$

$$CLR = \max\{p_{1_{max}}, p_{2_{max}}\} - \min\{p_{1_{min}}, p_{2_{min}}\} \tag{2.4e}$$

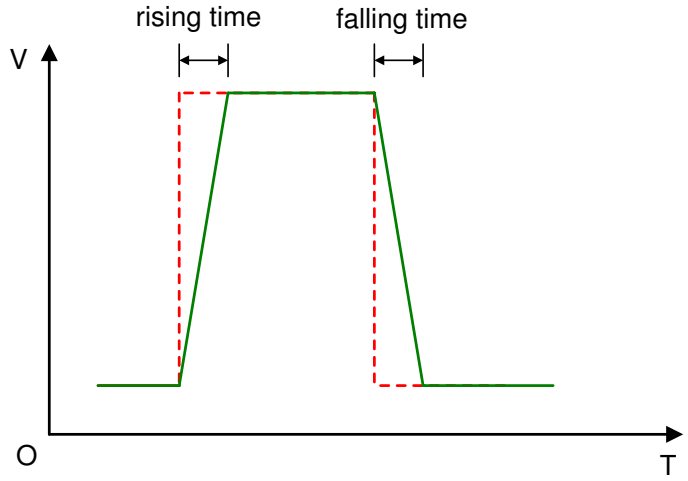
It can be concluded from the above equations that CLR represents both the deviation caused by delay models and the voltage variation.

2.2.3 Clock Slew Rate

The slew rate of an electronic circuit is generally defined as the maximum rate of change on the signal transmission. In physical design, we estimate the duration of the signal rising (falling) time for slew rate maintenance in the circuit. Signal rising (falling) time means the timing delay from low-level to high-level (high to low). For better signal quality therefore shorter rising

(falling) time, proper buffer insertion is necessary for driving power supply. An example of slew demonstration is shown in figure 2.5.

Figure 2.5 Slew effect on square wave.

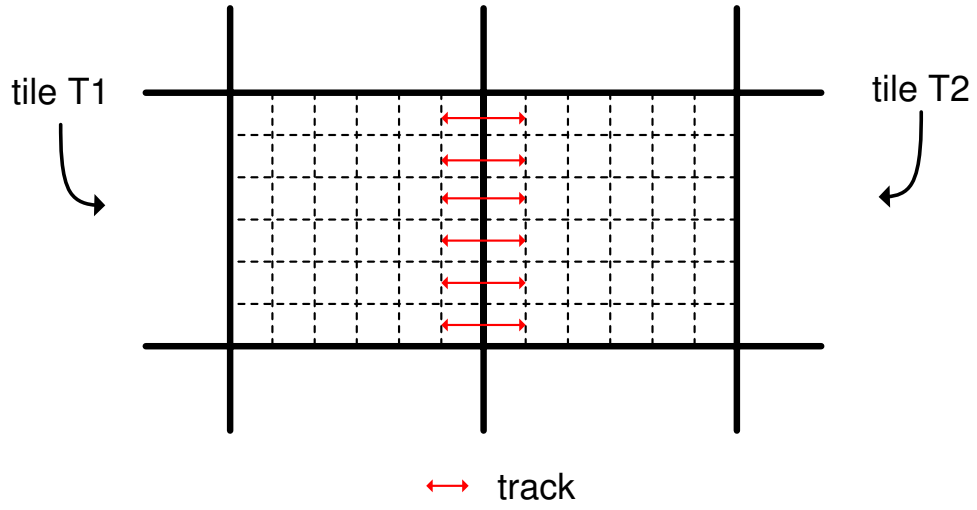


The restriction on clock slew describes the requirement on the rising (falling) signal rate. In some previous clock gating works [65, 11], it is defined to be the limited maximum load capacitance ($\leq 20 \times C_g$). In ISPD 2009 Clock Network Synthesis Contest [88, 39], it is defined to be the lasting time from 10% $\rightarrow$ 90% (90% $\rightarrow$ 10%) of the signal strength, and the upper limit is set to be 100 ps. Slew rate violation at any point would be regarded as a failure of the whole clock network synthesis.

2.2.4 Congestion Probability and Overflow

In congestion prediction, we model the degree of congestion for every tile in terms of the accumulated probability. Therefore, the congestion probability is the major metric for performance evaluation. In global routing, the maximal capacity for each edge is predefined, and the exceeding amount is named as overflow. We need to minimize the total overflow of the grid to improve the congestion. The whole chip area is decomposed into rectangular tiles, and $e_{i,j}$

Figure 2.6 The tracks between tile T_1 and tile T_2.



denotes the edge connecting the two adjacent tiles T_i and T_j. The respective capacity, demand and overflow on this edge are denoted as $cap_{i,j}$, $dem_{i,j}$ and $ovf_{i,j}$, respectively. ovf_{total} means the total overflow of the whole circuit. The capacity of an edge is determined by the number of tracks on it. As shown in figure 2.6, the red double-arrow line denotes a single track. Since there are six tracks in all on $e_{1,2}$, we have $cap_{1,2} = 6$. There will be overflow if the according demand of tracks $dem_{1,2}$ exceeds the capacity. The general definition of $ovf_{i,j}$ and ovf_{total} are shown as below.

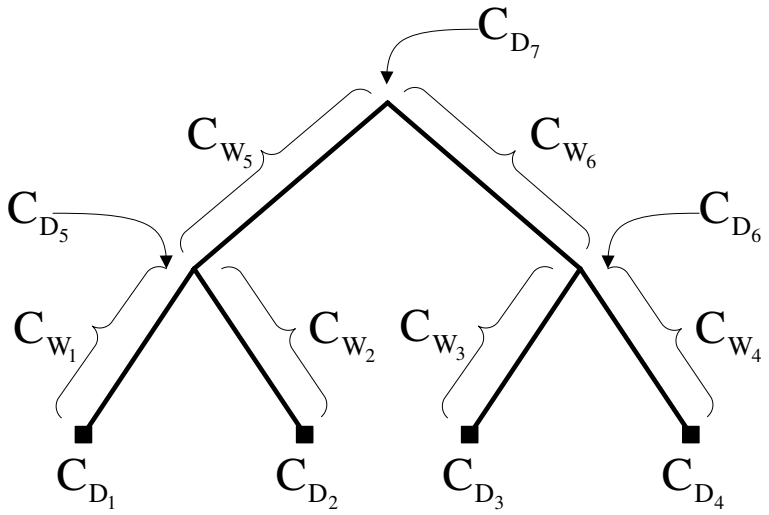
$$ovf_{i,j} = \begin{cases} dem_{i,j} - cap_{i,j} & : dem_{i,j} > cap_{i,j} \\ 0 & : dem_{i,j} \leq cap_{i,j} \end{cases} \tag{2.5}$$

$$ovf_{total} = \sum_{\text{all } e_{i,j}} ovf_{i,j} \tag{2.6}$$

2.2.5 Power on Capacitance and Wirelength

Power consumption has become a critical issue in the system-on-chip (SoC) design. Besides previous research works, much more work remains to be done

Figure 2.7 The capacitance accumulation in clock tree.



to automate the design flow. The equation for power usage computation is $P = \frac{1}{2}\alpha C_d f V_{dd}^2$. α is the amount of transition periods, C_d is the total load capacitance of the network, f represents the signal frequency and V_{dd} denotes the voltage supply. According to the above equation, the consumed power is in a linear relationship with the total capacitance C_d. Therefore, total capacitance is optimized in our work to reduce the power consumption.

In clock network synthesis, both the buffers and the wires have their contribution to the capacitance. We denote the procedure of capacitance accumulation in figure 2.7, where C_{D_i} denotes the downstream capacitance, and C_{W_i} denotes the capacitance contributed by a segment of wire. Notice that when there are buffer insertions, downstream capacitance will be changed according to the input capacitance of buffers.

$$C_{D_5} = (C_{D_1} + C_{D_2}) + (C_{W_1} + C_{W_2}) \tag{2.7a}$$

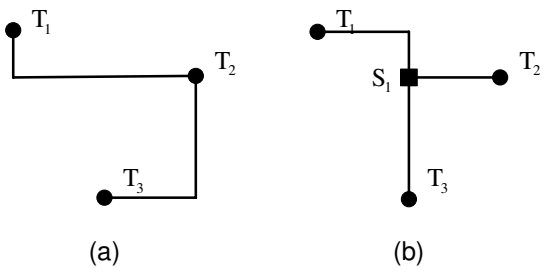
$$C_{D_6} = (C_{D_3} + C_{D_4}) + (C_{W_3} + C_{W_4}) \tag{2.7b}$$

$$C_{D_7} = (C_{D_5} + C_{D_6}) + (C_{W_5} + C_{W_6}) \tag{2.7c}$$

In global routing, the capacitance is approximated by the total wirelength.

We can reduce the power consumption by means of wirelength minimization. Notice that in our work, based on the rectilinear grid, the distance between each pair of pins is not the Euclidean distance. Instead, the Manhattan distance is applied which equals the sum of the horizontal and vertical distance. Rectilinear steiner minimum tree (RSMT) is an efficient approach to achieve the shortest connection of a network with multiple pins. Insertion of proper steiner points is a necessary step in the steiner tree construction. Two comparable examples are shown in figure 2.8, where T_1, T_2, T_3 are the three original pins to be connected, and S_1 is an additional steiner point. The minimum spanning tree (MST) solution is shown in the left case, and the rectilinear steiner minimum tree (RSMT) solution is shown in the right case. It is obvious that the steiner tree is a better solution with a shorter wirelength caused. The rectilinear steiner minimum tree problem is NP-hard, but plenty of approximation schemes have been proposed with acceptable performance [35]. Some are with even optimal performance at particular cases.

Figure 2.8 (a) Minimum spanning tree (b) Steiner tree.



Chapter 3

Literature Review

3.1 Overview

In this thesis, our research focus is located at the routability in VLSI physical design. During the early designing time, congestion prediction is applied in order to evaluate the degree of congestion in terms of respective probability on edge occupancy. In clock network synthesis, buffers cannot be inserted inside the blockage areas, which are preoccupied by macro blocks during the placement stage. Therefore, routability is maintained by proper buffer distribution for clock signal transmission. Clock gating design is an associating step of CNS aiming at power reduction. It is implemented by turn on/off active/inactive modules selectively. Subsequently except for the clock network, global routing implements the connection of the rest of the networks inside the chip area. Congestion and overflow are minimized and the according routability is maximized.

3.2 Congestion Prediction

By virtue of the non-stopping reduction of feature size in the VLSI design, interconnection delay replaces the transistor delay to be the dominant factor on

the signal delay. Therefore, routability has become a critical concern regarding the total timing performance of the circuit. However, improper design or operation in early stages will cause negative effect to the routing performance. For instance, arbitrary packing of modules will possibly lead to uneven distribution of congestion. As a result, the interconnection of such placement result may not be completed by global routers. Therefore, it is quite necessary to develop algorithms to improve routability during the placement or some other early designing steps, where congestion prediction is an effective technique of them. Congestion prediction can provide the probability of usage of all the tiles on the chip.

In retrospect to the past research achievements, several models have been proposed. In the works [14, 7, 57], a packing is divided into tiles and congestion is estimated in each tile, assuming that each net is routed in either L- or Z-shape. In [50], the congestion model used is the average net density on the boundaries of different regions in a floorplanning. In the papers [49, 54], probabilistic analysis is performed to estimate congestion and routability. They assume that all feasible routes have the same probability of being selected. In practice, routes of less bends are more desirable. In the papers [42, 96], extended versions of [54] are proposed. The authors take into account the impact of the number of bends in a routing path on the probability of occurrence of the path. However, the accuracy of their congestion models will depend on the accuracies of their predictions on the distribution of the number of bends in the routed circuit. The paper [99] proposed to predict congestion by using the Rent's rule. However, connections of the nets are already known in the floorplanning and placement stage, and we should be able to predict congestion more accurately than simply using the Rent's rule. The papers [94, 95] proposed to use global routers to estimate congestion, which will be more accurate but the runtime penalty is high. The paper [75] presented a probabilistic congestion prediction method based on routers intelligence. The paper [89]

gives an tutorial on all recent congestion technique and show the importance of the congestion prediction.

3.3 Clock Network Synthesis

After placement, the interconnection of all the networks should be completed in the stage of routing, in which clock network synthesis is the first step. In VLSI circuits, digital modules are synchronized by receiving clock signals, which are transmitted from a source through the clock network. In clock network synthesis, the clock skew is usually a major concern, which represents the difference of the signal arrival time at all the terminals. In order to synchronize one circuit, each terminal must be reached by the clock signal within a small time range. Otherwise, the performance on synchronization is unacceptable. Regarding the maximum attainable frequency of operation in the circuits, the resultant clock skew of the synthesis work must be reduced into a predetermined range.

Clock skew minimization is a popular research topic during the past decades. Plenty of research achievements have been proposed by previous researchers. Some earlier proposed works concentrated on the distribution of wirelength between the source and the sinks to achieve delay equalization. Jackson [40] firstly presented a clock routing algorithm in a top-down course. Later, a geometric perfect matching was proposed in [41] in a bottom-up procedure. More improvements were also made to reach exact equidistant tree [25] for the clock network. Afterwards, delay balancing using Elmore delay model [26] became prevalent to acquire more accurate information on the timing delay. Clock tree with exact zero skew [92] was proposed by applying such balancing method. The deferred-merging and embedding (DME) technique [3, 10] was proposed to achieve zero skew with a shorter wirelength in the clock tree. In topology generation, some algorithms were proposed for both un-buffered [24] and buffered [12] clock trees respectively. In [31, 17] buffer insertion was involved

for power supply and transition time reduction

In the recent years, tolerance on variation became a focus of attention. This is mainly due to the uncertainty of various uncertain factors inside a circuit, such as process [63, 100], voltage [76] and temperature [80]. In order to keep a chip stable and functioning well, methods with greater tolerance on variation are widely favored. Many researchers focused on robust algorithms for variation minimization. Techniques such as wire sizing [52], buffer sizing [15] and link insertion [71, 72] are applied. In ISPD 2009 clock network synthesis contest [88, 39], a voltage variation related objective, CLR, was formulated. Several new benchmarks were released accordingly. Subsequently, some related research works were proposed [86, 53, 55]. In [55], a dual-MST topology generation approach was devised and analyzed Its objective is to minimize the maximal edge cost during the weighted perfect matching.

3.4 Clock Gating Design

In modern VLSI design, optimization on the power consumption is necessary, in case chip overheat and battery shortage (for portable devices). Clock network affords the synchronization of digital modules in the circuits. However, it is also quite power consuming. From statistics, it is shown that twenty to fifty percent of the dynamic power usage is contributed by the clock network [46]. Therefore, optimization on clock network is an effective way to make the chip less power consuming. At each clock period, only a small portion of the digital module set is active. The remaining modules are temporarily idle and they can be turned off for power saving. Based on this assumption, gated clock network is proposed in the sequential circuits for enable/disable control signal transmission. The principal idea is to turn off the idle modules and the according sections of the clock tree, and the unnecessary switching power can thus be cut down.

Clock gating can be applied on logic level [8], register-transfer-level level [22]

and architecture level [56], respectively. Nevertheless, optimization on these upper design levels may ignore the physical information and cause unnecessary detours or snaking wires. Upper level improvement may be offsetted by lower level implementation [30]. As a matter of fact, physical location of the modules should be taken into account during the clock gating design, in case wirelength overhead thus power waste.

Some achievements have been proposed with simultaneous logical and physical concerns. The design of activity-driven gated clock network was proposed in [91] and [27]. However, clock skew was only concerned by gate number balancing, and the contribution of wires were ignored. Meanwhile, the gate insertion was not applied concurrently within the topology generation. In [13], similarity of activity patterns between clock nodes was utilized, and the performance was improved with reduced power consumption. Instruction stream was proposed in [64] to simulate the probabilistic information of each module. In [65], a gating method regarding microprocessor design was developed to minimize the switched capacitance. However, no relationship of activities was concerned while merging each pair of nodes. Meanwhile, the resulted clock tree was still non-zero skew. Recently, a comprehensive technique was proposed in [11]. It is composed of a recursive computation on effective switched capacitance and a solution sampling method based on merging segment set. Despite the improvement on performance, the approach is very time and memory consuming, and the slew rate is too ideal (limit on downstream capacitance) which may not be applicable for industry use.

3.5 Global Routing

After clock network synthesis, global routing and detailed routing are applied to complete the interconnections of the rest of the networks. In the stage of

global routing, the grid is decomposed into global bins. By such decomposition, all the pins are labelled with the coordinates of their according global bins. In each network, only the pins from different global bins need to be connected together. The specific inner-bin connection will be implemented in the detailed routing. Plenty of research works regarding global routing have been proposed with a great deal of improvements achieved [60]. Meanwhile, there were two global routing contests held in 2007 [62, 37] and 2008 [38], which greatly stimulated the research interests in global routing.

Among those state-of-the-art global routers, they generally employ two categories of basic techniques: (1) iterative rip-up and rerouting (2) multi-commodity flow. The approach of iterative rip-up and rerouting does not confine routing selections in the early steps. However, it completes the task by means of numerous iterations of rerouting to gradually approach the final target of zero overflow. Repetitious rerouting would cost additional runtime to fulfill the work, thus focus is usually centralized on those congested area to make the processing efficient. The distribution of routing power is asymmetrical among all the nets, and congested nets usually get a big share. There are already a number of algorithms proposed based on it. Chi Dispersion [32] is a global routing approach implemented based on cost amplification. Labyrinth [44] is implemented based on the scheme of predictable routing. These are the two early global routers employing this technique. Recently, more and more global routers are presented, including Archer [68], BoxRouter [18], FGR [74], FastRoute [69, 70, 101], MaizeRoute [58, 59], NTHU-Route[29, 9], NTUgr [33] and NCTU [21]. Most of them apply FLUTE [19, 20], a rather efficient tool based on look-up table, to generate the Rectilinear Steiner Minimum Tree (RSMT) for each network. FastRoute is of rather high speed and it can be integrated into placement to provide interconnection information. MaizeRoute is implemented with extreme-edge-shift technique, and NTHU

uses the concept of region-based constraint for overflow reduction. For the approach of multi-commodity flow, the global routing problem is formulated and transferred into an integer linear programming (ILP) problem to be solved. A recent router from the work by Albrecht [1] provides an approximation method based on this approach.

3.6 Summary

From the above review, it is shown that some achievements have been made by previous researchers on the topic of routability. Nevertheless, owing to the development of the craft on semiconductors and the enlargement of the integration scale, new requirements on routability have been proposed. Therefore, there are still large space for performance improvement. In congestion estimation, a lot of efficient models have been proposed, such as some predefined patterns for resource assignment and probabilistic analysis. The influence caused by routing procedure still lacks enough concerns. In clock network synthesis and clock gating design, equidistant clock tree and zero skew approach have been developed and improved. Nonetheless, excessive focus on wirelength optimization would deteriorate the tolerance on variability and reduce the robustness of the work. In global routing, recent global routers focused on fast reduction of overflow, but the optimization on wirelength and via connection is partly ignored.

Chapter 4

Congestion Prediction

4.1 Overview

Congestion prediction is of crucial importance in the early stages of the physical design. In this chapter, we propose three congestion models: SMD (shortest Manhattan distance), Detour model [84] and 3-step approach [83]. From the experimental results it is shown that the 3-step approach is the most efficient one considering the accuracy of congestion prediction.

4.2 Problem Formulation

Congestion modeling is an important part of interconnect estimation during the floorplanning and placement stage. Given a packing which is partitioned into $l_h \times l_w$ tiles (according to the length of tile, t_l), we will calculate the net density at each tile according to the congestion model. Net density means the accumulated probability of tracking paths in each tile, which are contributed by all the nets in the chip. Based on the congestion information obtained from the net density, we can evaluate the routability of the packing result. The objective of our work is to minimize the difference between the results of estimation and routing, in order to predict the grade of congestion accurately at early stages of the physical design.

Table 4.1 Notations in congestion prediction

Notation	Description
t_l	Length of a tile
c^h_{max}	Maximum horizontal wire capacity inside a tile
c^v_{max}	Maximum vertical wire capacity inside a tile
$c_k(r)$	The number of tiles that is r tiles from the source of net k
DT_k	The shortest Manhattan distance between the source and sink of the net k
$d_k(x, y)$	The distance from the source of net k to tile (x, y)
$P_k(x, y)$	A rough estimation of the probability of net k passing through the tile (x, y)
$W(x, y)$	The weight of tile (x, y)
CF_k	Congestion factor of the net k
l^k_d	Detoured length of the net k
$E_k(x, y)$	Probability of net k passing through (x, y)
$E^h_k(x, y)$	Probability of net k passing through (x, y) horizontally
$E^v_k(x, y)$	Probability of net k passing through (x, y) vertically
$E^h(x, y)$	Expected number of wires passing through (x, y) horizontally
$E^v(x, y)$	Expected number of wires passing through (x, y) vertically
$A^h(x, y)$	Actual number of wires passing through (x, y) horizontally, obtained from global routing result
$A^v(x, y)$	Actual number of wires passing through (x, y) vertically, obtained from global routing result
(s^x_k, s^y_k)	Coordinate of the source tile of net k
(t^x_k, t^y_k)	Coordinate of the terminal tile of net k
T^d_k	The set of extra tiles when the detoured nets pass through outside the bounding box of net k
T_k	The set of tiles inside the bounding box of net k
$T_k(d)$	The set of tiles inside the bounding box of net k, being d tiles away from the source

4.3 Analysis of Congestion Models

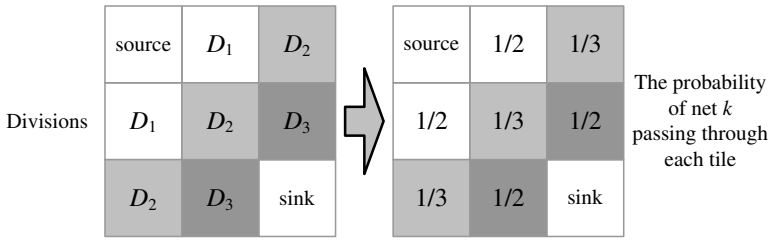
In the previous works regarding congestion prediction, we usually break down the multi-pin nets into 2-pin nets first. A network with N pins can be decomposed into $N-1$ subnets, each subnet is a connection of two pins. In this section, we mainly propose three models of congestion prediction. All of them are constructed based on 2-pin subnets decomposition. Respective principal analysis and performance comparison are listed. These models are SMD, Detour and 3-step approach, which are discussed as follows.

4.4 SMD Model

When we assume that all the nets are routed in their shortest Manhattan distances, the tiles within the smallest bounding box of net k can be divided into $DT_k - 1$ divisions where DT_k is the shortest Manhattan distance between the source and sink of net k. An example is shown in figure 4.1. The tiles are divided into three divisions D_1, D_2 and D_3. Intuitively, if the nets are restricted to be routed within the bounding box with the shortest Manhattan distance, the nets will pass through exactly one tile in each division of the corresponding smallest bounding box. Instead of assuming that the probability of each possible route is the same, we propose a new congestion model, SMD model, assuming that a net will pass through the tiles in the same division with the similar probability. Thus, all the tiles that have the same distance from the source or sink of a net k will have the same probability of being passed through by net k.

Let $s_k(x, y)$ denote the distance from the source of net k to tile (x, y). The tiles having the same distance from the source of net k will be grouped in the same division. Let $c_k(r)$ be the number of tiles that is r tiles from the source of net k. Hence, the probability of net k passing through (x, y), $P_k(x, y)$, can

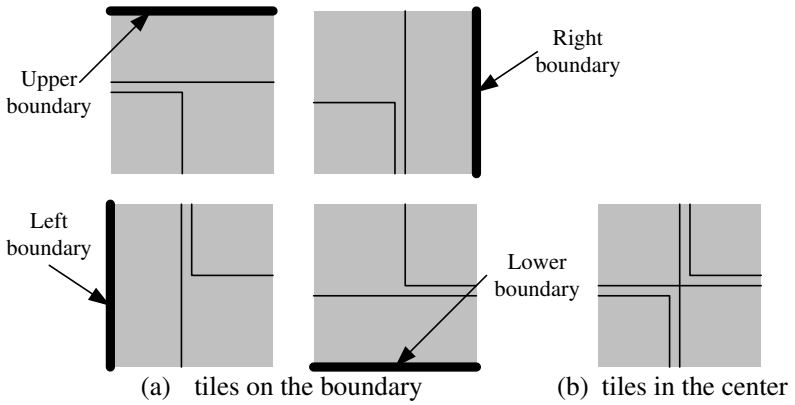
Figure 4.1 SMD model for a two-pin net



be calculated by the following equation:

$$P_k(x, y) = \frac{1}{c_k(d_k(x, y))} \tag{4.1}$$

Figure 4.2 Possible routes inside a tile (routed from the upper-left corner to the lower-right corner)



In addition, a net may pass through a tile either horizontally or vertically. When a net is routed from the upper-left corner to the lower-right corner of the bounding box, the net may pass through a tile with a path as shown in figure 4.2. If the tile is on the boundary of the bounding box, the route may pass through the tile in two ways. The four different cases of the tile lying along the top, the left, the bottom and the right boundary are shown in figure 4.2a. If the tile is on the left or right (at the top or bottom) of the bounding box,

the length of the route passing through the tile horizontally (vertically) is $\frac{0.5t_l}{2}$ and the length of the route passing through the tile vertically (horizontally) is $\frac{1.5t_l}{2}$. If a tile is not on the boundary of the bounding box, the net may pass through the tile in four different ways. They are shown in figure 4.2b. In this case, the length of the route passing through the tile horizontally or vertically is $\frac{2t_l}{4}$. Thus, we can calculate $E_k^h(x,y)$ and $E_k^v(x,y)$ by the following equations:

$$E_k^h(x,y) = \begin{cases} \frac{3\times E_k(x,y)}{4} & : y = s_k^y \text{ or } y = t_k^y (x \neq s_k^x \text{ and } x \neq t_k^x) \\ \frac{E_k(x,y)}{4} & : x = s_k^x \text{ or } x = t_k^x (y \neq s_k^y \text{ and } y \neq t_k^y) \\ \frac{E_k(x,y)}{2} & : \text{others} \end{cases} \tag{4.2a}$$

$$E_k^v(x,y) = \begin{cases} \frac{E_k(x,y)}{4} & : y = s_k^y \text{ or } y = t_k^y (x \neq s_k^x \text{ and } x \neq t_k^x) \\ \frac{3\times E_k(x,y)}{4} & : x = s_k^x \text{ or } x = t_k^x (y \neq s_k^y \text{ and } y \neq t_k^y) \\ \frac{E_k(x,y)}{2} & : \text{others} \end{cases} \tag{4.2b}$$

Finally, the expected number of wires passing through (x,y) horizontally and vertically, $E^h(x,y)$ ($E^v(x,y)$), can be calculated by following equations:

$$E^h(x,y) = \sum_{\text{all net } k} E_k^h(x,y) \tag{4.3a}$$

$$E^v(x,y) = \sum_{\text{all net } k} E_k^v(x,y) \tag{4.3b}$$

4.5 Detour Model

The congestion models discussed before assume that all nets are routed in their shortest Manhattan distances. However, the congestion model can be more accurate when we consider net routing without this assumption. This is reasonable as it is nearly impossible to route all the nets in their shortest

Manhattan distances for any large circuit. In this section, we will also propose a new congestion model (called Detour model [84]) in which each net is not necessarily routed in its shortest Manhattan distance. Detour model is proposed based on the SMD model. It assumes that a route may have detours.

4.5.1 Estimation of Detoured length

Table 4.2 Percentage of detoured nets

Test Cases	Percentage of detoured nets (%)		
	$R1$	$R2$	$R3$
hp	11.232	8.664	6.432
apte	17.75	16.363	10.713
ami33	8.794	6.432	3.804
ami49	15.747	10.633	5.452
playout	0.389	0.256	0.081

In this section, we show how to estimate the congestion of the tiles when nets may detour. Obviously, the nets may detour only when some locations of the packings are very congested. We perform global routing on the packings to illustrate the percentage of detoured nets under different routing environments. The experimental results are shown in table 4.2. We can see that if the packing is more congested, more nets may detour. Thus, we can first use SMD model to evaluate the congestion of the packing and calculate the congestion factor of each net k by following equation:

$$CF_k = \sum_{(x,y)\in T} \frac{2 \times (E^h(x,y) - E^h_k(x,y) + E^v(x,y) - E^v_k(x,y))}{|s^k_x - t^k_x| + 1 \times |s^k_y - t^k_y| + 1} \times (c^h_{max} + c^v_{max}) \tag{4.4}$$

where $s^k_x \leq x \leq t^k_x$ and $s^k_y \leq y \leq t^k_y$.

From the equation, we can see that if CF_k is larger than one, it means that the bounding box is very congested for net k and the net k may likely detour.

Thus, we can estimate the length of detour of net k by the following equation (if l_d^k is smaller than zero, it is adjusted to zero):

$$l_d^k = \lfloor (CF_k - 1) \times DT_k \rfloor \tag{4.5}$$

Table 4.3 Improvement of wirelength estimation

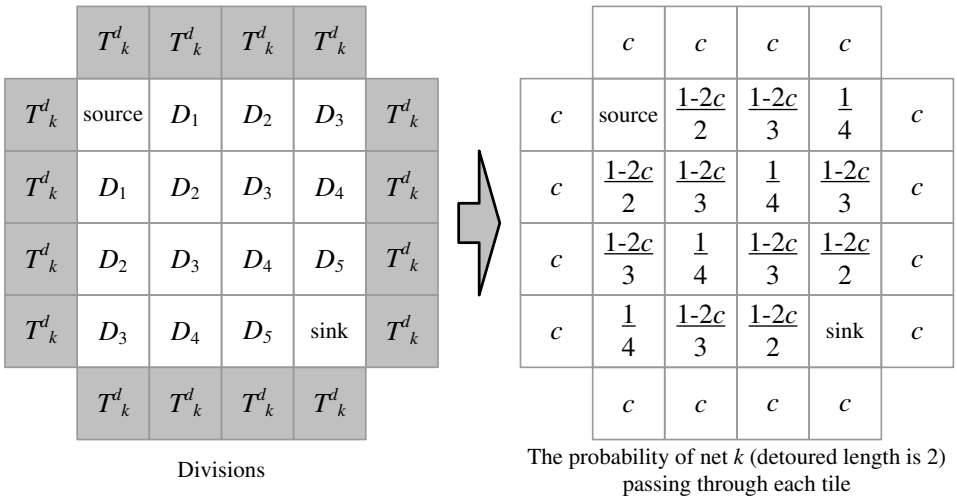
Test Cases	Actual wirelength	Estimated wirelength	RMST
hp	1091.10(0.00%)	1079.44(1.56%)	1068.80(2.18%)
apte	1888.24(0.00%)	1873.92(1.01%)	1846.60(2.3%)
ami33	2065.78(0.00%)	2054.18(0.61%)	2039.90(1.25%)
ami49	3127.44(0.00%)	3028.04(3.17%)	2940.30(5.38%)
playout	12307.02(0.00%)	12305.78(0.06%)	12304.50(0.06%)

We have performed global routing on the packings to verify the estimation of detoured length. The experimental results are shown in table 4.3. From the experiments, we can see the error between our estimated wirelength (the multi-pin nets are decomposed into 2-pins nets by rectilinear minimum spanning tree first) and the actual wirelength by global routing is always smaller than the error between the wirelength of rectilinear minimum spanning tree (RMST) and the actual wirelength. It means that we can always estimate the wirelength more accurately by using congestion factor.

4.5.2 Congestion Estimation

As the detoured nets may pass through outside the bounding box, some extra tiles may be passed through by those nets. These extra tiles are the tiles when the detoured nets pass through outside the bounding box of net k with detoured length less than l_d^k. First, we define T_k^d as those extra tiles for a net k. In real cases, $P_k(x, y)$ for $(x, y) \in T_k^d$ will depend on the location of the congestion area. Detours happen because of some random over-congested tiles. Because of this random situation, it is reasonable to assume that all tiles

Figure 4.3 Detour model for a two-pin net



(x, y) in T_k^d have the same probability. Hence, we can build the congestion map based on SMD model. An example is shown in figure 4.3.

The detoured net may often pass through outside the bounding box with less bends. If edge shifting [69] is used, the most common case is that either the whole vertical or horizontal path is outside the bounding box. Thus, either all horizontal segments or all vertical segments are passing through outside the bounding box. In this case, we assume that half of the wirelength of the detoured net will be contributed by the tile in T_k^d.

The detoured net may often pass through outside the bounding box with less bends. The most common case is that either the whole vertical or horizontal path is outside the bounding box. Thus, we assume that half of the wirelength of the detoured net will be contributed by the tile in T_k^d. Hence, we can calculate $P_k(x, y)$ where $(x, y) \in T_k^d$ by following equation:

$$E_k(x, y) = \frac{DT_k + l_d^k}{2 \times |T_k^d|} : (x, y) \in T_k^d \tag{4.6}$$

where $|T_k^d|$ is the number of tiles in T_k^d

Hence, we can calculate $E_k(x, y)$ of other tiles (for any $(x, y) \notin T_k^d$) that inside the bounding box following equation:

$$E_k(x, y) = \frac{1 - c * |T_k^d|_{s_k(x,y)}}{c_k(s_k(x, y))} \tag{4.7}$$

where $|T_k^d|_{s_k(x,y)}$ is the number of tiles in same diagonal of the division $s_k(x, y)$ and c is $E_k(x, y)$ for any $(x, y) \in T_k^d$. In addition, a net may pass through a tile either horizontally or vertically. We will calculate $E_k^h(x, y)$ and $E_k^v(x, y)$ by equation 4.2.

4.6 3-Step Approach

The estimation process is divided into three steps: preliminary estimation, detailed estimation and congestion redistribution. To avoid over-estimating congestion, we perform a preliminary estimation step first to determine which regions are likely to be over-congested. A region should be more attractive to net routing if it is less congested. Then, we will make use of this information to predict the congestion measures during the detailed estimation step. We use a SMD model described in section 4.4 because of its simplicity and experimental results have shown that this model can give accurate estimations. Finally, congestion redistribution will be performed to simulate the rip-up and reroute operations of the detailed routing step by moving wires from over-congested regions to less congested regions. We use a 3-step approach [83] as follows:

- Preliminary Estimation: We estimate the congestion measure at each tile roughly according to the bounding box of each net so that we can determine which regions are likely to be over-congested.

- Detailed Estimation: Based on the information obtained from the preliminary estimation step, we estimate the congestion measure at each tile by using a diagonal-based congestion model.

- Congestion Redistribution: We will simulate the rip-up and reroute process of the routing stage by moving wires from over-congested tiles to less congested tiles.

4.6.1 Preliminary Estimation

In practice, we will choose to route a net over the tiles that are less congested to prevent overflow. It means that some tiles are more attractive to net routing and some tiles are less. However, this fact is usually ignored in traditional congestion models. In our approach, a preliminary estimation of the congestion map will be performed to obtain this information. If a rough estimation of the congestion measure of a tile, $P(x, y)$, is above the maximum wire capacity, the tile (x, y) will be less attractive to net routing. On the other hand, if $P(x, y)$ is well below the maximum wire capacity, the tile (x, y) will be more attractive to net routing. We will make use of these $P(x, y)$ values to improve the accuracy of the detailed estimation step.

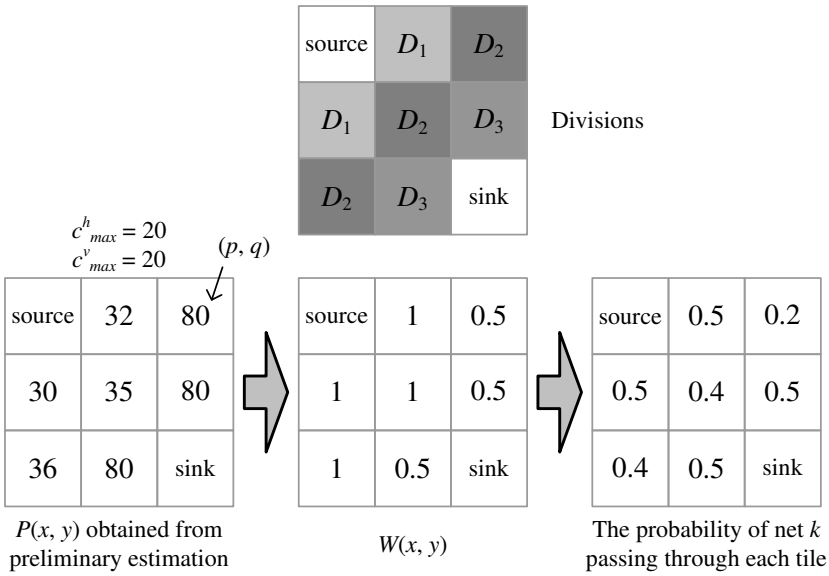
In this preliminary estimation step, we assume that all the tiles inside the bounding box of a net k, T_k, have the same probability, $P_k(x, y)$, of being passed through by net k. In addition, we assume that the nets can be routed in their shortest Manhattan distances. The wirelength and the area of the bounding box can be computed as $|t_k^x - s_k^x| + |t_k^y - s_k^y| + 1$ and $(|t_k^x - s_k^x| + 1) \times (|t_k^y - s_k^y| + 1)$ respectively. $P_k(x, y)$ can thus be calculated by the following equation:

$$P_k(x, y) = \frac{|t_k^x - s_k^x| + |t_k^y - s_k^y| + 1}{(|t_k^x - s_k^x| + 1) \times (|t_k^y - s_k^y| + 1)} \tag{4.8}$$

We can then obtain a preliminary estimation by adding up the congestion measures due to different nets:

$$P(x, y) = \sum_{\text{all } k} P_k(x, y) \tag{4.9}$$

Figure 4.4 An example of computing the congestion measures for a two-pin net in the detailed estimation step



4.6.2 Detailed Estimation

In our approach, we will predict the congestion measures by using a diagonal (orthogonal to the source-sink connection) based model during the detailed estimation step. We first assume that all the nets are routed in their shortest Manhattan distances. The tiles inside the smallest bounding box of net k can be divided into $DT_k - 1$ divisions where DT_k is the shortest Manhattan distance between the source and the sink. An example is shown in figure 4.4.

In this example, the tiles are divided into three divisions D_1, D_2 and D_3. Intuitively, if the net is restricted to be routed within the bounding box, the net will pass through exactly one tile in each division. We assume that the net will pass through the tiles in the same division with probabilities weighted according to $W(x, y)$ where $W(x, y)$ is computed by the following equations according to the $P(x, y)$ obtained in the preliminary estimation step.

$$W(x,y) = \begin{cases} 1 & : P(x,y) < (c^h_{max} + c^v_{max}) \\ \frac{c^h_{max}+c^v_{max}}{P(x,y)} & : \text{otherwise} \end{cases} \tag{4.10}$$

If $P(x,y)$ is smaller than the sum of c^h_{max} and c^v_{max}, the tile (x,y) is unlikely to be over-congested and so $W(x,y)$ is 1. If $P(x,y)$ is larger than the sum of c^h_{max} and c^v_{max}, that tile should have a smaller $W(x,y)$ when $P(x,y)$ is larger. It reflects the case in the routing stage that the nets will be routed to pass through less congested tiles. Hence, the probability of net k passing through (x,y), $E_k(x,y)$, can be calculated according to the weight of each tile, $W(x,y)$, by the following equation:

$$E_k(x,y) = \frac{W(x,y)}{\sum_{(i,j)\in T_k(d_k(x,y))} W(i,j)} \tag{4.11}$$

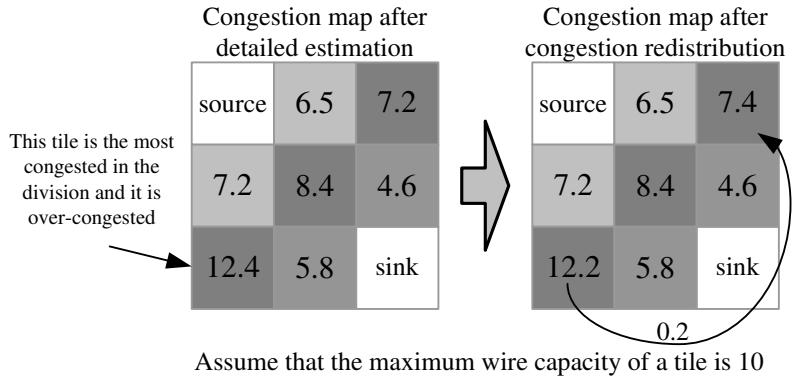
In the example of figure 4.4, c^h_{max} and c^v_{max} are 20. When we focus on division D_2, (p,q) the tile at the upper right corner is a congested tile according to the preliminary estimation step because $P(p,q)$ is 80, which is larger than the sum of c^h_{max} and c^v_{max}. Thus, $W(p,q)$ should be smaller than 1 and it is computed as 0.5 according to equation 4.10. Hence, the probability of net k passing through (p,q), $E_k(p,q)$, is 0.2. It is smaller than the others in the same division because the tile (p,q) is likely to be over-congested. In addition, a net may pass through a tile either horizontally or vertically. We will calculate $E^h_k(x,y)$ and $E^v_k(x,y)$ by equation 4.2.

4.6.3 Congestion Redistribution

In real routing, if some tiles are over-congested or some nets cannot be routed, rip-up and reroute will be performed. In our approach, we perform congestion redistribution to achieve the same purpose of moving wires from over-congested tiles to less congested tiles. We will only move around those congestion measures within the same diagonal (division). In this case, we may simulate the

routing process when less congested path is always selected and the total congestion of one division can be maintained at one. An example is shown in figure 4.5. In $T_k(1)$ of net k, the tile with congestion estimation 7.2 is the most congested in this division but it is not over-congested (less than the maximum wire capacity of a tile). Thus, no action will be taken. In $T_k(2)$, the tile with 12.4 is the most congested in this division and is over-congested. Thus, we will move 0.2 (net k's contribution to the congestion measure of this tile) from this tile to the least congested tile of the same division.

Figure 4.5 An example of congestion redistribution



In general, we will find the tile, (x_m, y_m), with the maximum vertical (horizontal) congestion and the tile, (x_l, y_l), with the minimum vertical (horizontal) congestion from each division of all the nets. If the tile with the maximum vertical (horizontal) congestion is over-congested, we will move $E_k^v(x_m, y_m)$ ($E_k^h(x_m, y_m)$) from (x_m, y_m) to (x_l, y_l). After redistribution, the summation of $E_k^v(x, y)$ ($E_k^h(x, y)$) in the same division still equals one. Thus, the assumption that each net will pass through exactly one tile in each division within the bounding box still holds.

4.7 Experimental Results

In the experiments, the test cases used are the ISPD-02 suite circuits [36]. The length of a tile, t_l, is $40\mu m$. The detailed information of the testing circuits are shown in table 4.4. Our prediction models are robust approaches with no inclusion of circuit-oriented parameter tuning. Therefore, the performance of our work should be consistently good based on the packing results from various placers. In our experiments, all the circuits are first placed using a wirelength driven placer, Capo [5]. Four placement solutions are obtained for each test case. Global routing is then performed on each placement solution by a maze routing based global router [44]. During global routing, we set wiring capacity value to simulate two environments: more congested and less congested cases. For the data sets shown in table 4.5, there are about $0\% - 2\%$ over-congested tiles after global routing. Different congestion models are then used to estimate the congestion of the placed circuits and their estimations are then compared with the actual congestion measures obtained from the global router.

Table 4.4 Information of the test cases

Test Cases	No. of Cells	No. of Nets	No. of 2-pin Nets	No. of Tiles
*ibm*01	12506	14111	36455	57 × 57
*ibm*02	19342	19584	61615	82 × 82
*ibm*03	22853	27401	66172	89 × 87
*ibm*04	27220	31970	73889	85 × 86
*ibm*05	28146	28446	97862	60 × 60
*ibm*06	32332	34826	93366	81 × 82
*ibm*07	45639	48117	127522	97 × 97
*ibm*08	51023	50513	154377	104 × 103
*ibm*09	53110	60902	161186	118 × 118
*ibm*10	68685	75196	222371	194 × 189
*ibm*11	70152	81454	199332	130 × 129
*ibm*12	70439	77240	240520	171 × 171
*ibm*13	83709	99666	257409	141 × 141
*ibm*14	147088	152772	394044	151 × 151
*ibm*15	161187	186606	529215	170 × 169
*ibm*16	182980	190048	588775	204 × 203
*ibm*17	184752	189581	670455	182 × 182
*ibm*18	210341	201920	617777	163 × 163

We compare our SMD model, Detour model and the 3-step approach with

Table 4.5 Comparison on the mean and standard deviation of error of the congestion models for more congested circuits

Test Cases	c^h_{max} and c^v_{max}	Lou's Model		Westra's Model		SMD Model		Detour Model		3-step Approach	
		μ	μ_{std}	μ	μ_{std}	μ	μ_{std}	μ	μ_{std}	μ	μ_{std}
*ibm*01	27	17.36	10.01	15.06	10.05	14.32	9.57	14.05	9.58	12.63	9.45
*ibm*02	46	16.96	11.27	14.12	11.32	11.63	10.88	11.20	10.92	10.15	10.64
*ibm*03	45	32.34	10.78	27.59	10.75	24.29	10.82	22.32	11.05	20.66	10.78
*ibm*04	140	4.51	11.69	4.37	11.64	4.12	11.02	4.12	10.67	4.10	10.57
*ibm*05	70	27.21	11.23	23.27	11.37	18.28	11.19	18.10	11.54	14.22	10.67
*ibm*06	47	23.43	12.32	21.20	12.15	17.61	12.35	17.05	12.02	14.65	12.64
*ibm*07	53	13.51	12.58	12.05	12.43	10.44	11.65	10.33	11.98	9.44	12.21
*ibm*08	400	2.17	11.06	2.07	11.02	1.86	10.43	1.85	10.54	1.85	9.88
*ibm*09	50	15.76	10.78	13.62	10.68	11.43	10.97	11.32	11.43	10.34	12.08
*ibm*10	50	18.16	12.63	11.71	12.73	9.13	12.22	9.02	12.05	8.36	11.54
*ibm*11	55	14.38	12.43	11.55	12.62	10.28	11.97	10.01	11.54	9.43	12.64
*ibm*12	60	20.79	11.57	14.86	11.67	11.42	11.64	11.12	11.95	10.16	12.67
*ibm*13	60	14.67	12.12	11.97	12.69	10.21	11.00	9.98	11.34	9.35	10.68
*ibm*14	65	13.38	11.67	10.52	11.09	9.88	10.97	9.57	11.68	9.32	11.02
*ibm*15	80	17.19	10.52	12.06	10.67	10.33	11.02	10.10	10.57	9.28	10.64
*ibm*16	60	19.54	12.98	14.47	12.84	11.58	12.54	11.26	12.06	10.58	12.03
*ibm*17	80	18.56	10.67	12.70	10.67	10.57	9.87	10.23	9.77	9.69	9.44
*ibm*18	70	17.20	12.04	14.56	11.97	12.76	12.54	12.35	12.44	11.65	12.22
Average		17.06	11.58	13.76	11.58	11.67	11.26	11.33	11.29	10.33	11.21
w.r.t. Lou's		1.00	1.00	0.81	1.00	0.68	0.97	0.66	0.97	0.61	0.97

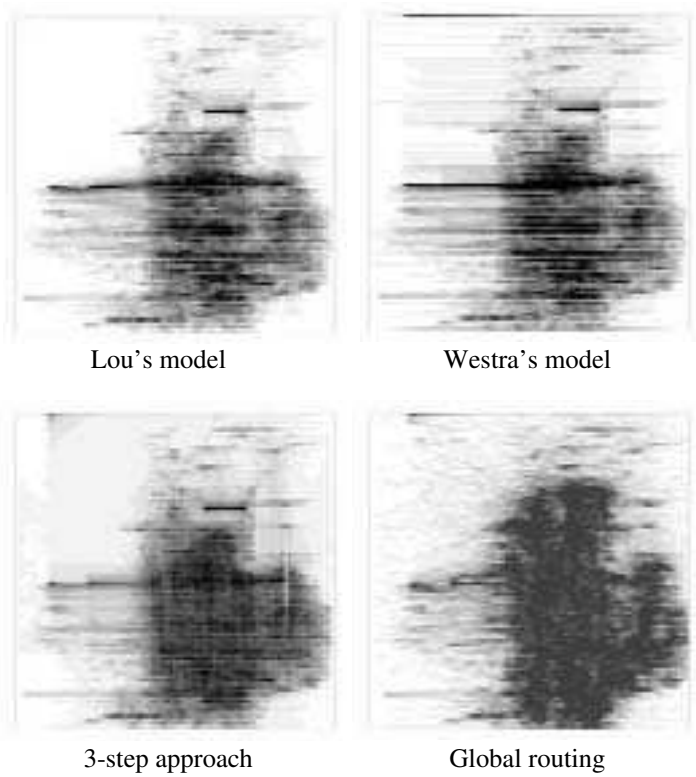
the models from Lou's model [54] and Westra's model [96]. We have implemented all the congestion models and compared the estimations with the results of the maze router. All programs were written in the C language and run on a machine (Sun Blade 1000) with $750MHz$ processor and $2GB$ memory. We will compare the congestion models by calculating the mean of error, μ and the standard deviation of error, μ_{std} according the following equations:

$$\mu_h = \frac{\sum_{(x,y)\in T} \frac{|A^h(x,y) - E^h(x,y)|}{c^h_{max}}}{|T|}$$

$$\mu_v = \frac{\sum_{(x,y)\in T} \frac{|A^v(x,y) - E^v(x,y)|}{c^v_{max}}}{|T|}$$

$$\mu = \frac{\mu_h + \mu_v}{2} \tag{4.12}$$

Figure 4.6 Congestion maps of horizontal wires (case: ibm03)



$$\mu_{std} = \sqrt{\frac{\sum_{(x,y)\in T}((\frac{|A^v(x,y)-E^v(x,y)|}{c^v_{max}} - \mu)^2 + (\frac{|A^h(x,y)-E^h(x,y)|}{c^h_{max}} - \mu)^2)}{|T|}} \quad (4.13)$$

where T is the set of all tiles that either their actual congestion measures or estimated congestion measures are non-zero.

The experimental results are shown in table 4.5. The values are the averages of the four placement solutions for each test case. We can see that SMD model can give smaller means in most cases than the Lou's [54] and Westra's [96] models. Detour model also have smaller standard deviations of error than SMD model but the improvement is not significant. The accuracies can be further improved when we can simulate the rip-up and reroute operations by performing the preliminary estimation and congestion redistribution steps by the 3-step approach. For the standard deviation of the error, the results of all

Figure 4.7 Error distribution of horizontal wires (case: ibm03)

Lou's model Westra's model 3-step approach

the models are similar.

In figure 4.6, the congestion maps obtained by different congestion models and the actual one (obtained by global routing) are shown. We can see that there are many regions that are predicted as over-congested in the Lou's and Westra's models and there are also a lot of empty regions in their models. However, the nets can be ripped up and rerouted to avoid passing through the over-congested regions. There is thus no over-congested region after global routing and most of the tiles in the placed region are used by some nets. In our modeling, we applied the preliminary estimation and congestion redistribution steps, and a similar congestion map can be obtained. Clearer comparisons can be illustrated by the error distributions of different congestion models in figure 4.7. We can see that differences occur in the surroundings of the over-congested tiles. It is because the global routing step will rip up the nets from the over-congested tiles and reroute them in the less congested tiles in the surroundings. Results show that we can improve the congestion estimation accuracy in different parts of the circuit.

In addition, we have compared the runtime of different congestion models. The results are shown in table 4.6. If we apply the detailed estimation step only, the runtime is faster than both the Lou's [54] and Westra's [96] models. If

Table 4.6 Comparison of the runtime of the congestion models

Test Cases	Lou's Model (s)	Westra's Model (s)	SMD Model (s)	Detour Model (s)	3-step Approach (s)	Global Routing [Kastner et al.] (s)
*ibm*01	0.24	0.14	0.13	0.19	0.31	190
*ibm*02	0.46	0.28	0.27	0.42	0.60	454
*ibm*03	0.92	0.58	0.53	0.92	1.10	987
*ibm*04	0.94	0.54	0.50	0.93	1.00	806
*ibm*05	1.03	0.60	0.55	0.94	1.20	1058
*ibm*06	0.64	0.35	0.34	0.59	0.77	642
*ibm*07	1.16	0.87	0.67	1.08	1.44	1206
*ibm*08	1.46	1.00	0.94	1.53	1.96	2021
*ibm*09	1.50	1.02	0.95	1.43	2.02	2217
*ibm*10	5.09	4.56	4.20	6.23	7.93	8820
*ibm*11	2.25	1.61	1.51	2.49	3.07	3021
*ibm*12	6.00	5.49	5.08	8.03	9.60	10543
*ibm*13	2.93	2.05	1.90	2.98	3.91	4680
*ibm*14	4.45	3.14	2.94	4.87	5.93	9480
*ibm*15	6.99	5.51	5.11	7.61	10.21	14220
*ibm*16	8.01	6.32	5.90	9.39	11.68	15684
*ibm*17	9.77	7.81	7.27	11.36	14.24	20547
*ibm*18	4.84	3.14	2.92	4.95	6.43	11235
Ave.	3.26	2.50	2.32	3.67	4.63	5990

we also apply the preliminary estimation and congestion redistribution steps, the runtime is slower. However, it is still acceptable because a more accurate congestion model can help us to spend less time in the later routing stage.

Finally, we compare the mean of error of the congestion models with ISPD-07 suite circuits [37] and the circuits are global routed by two different global routers (AMGR [98] and MaizeRouter [58]). The results are shown in the table 4.7 and table 4.8. We can see that the 3-step approach also have smallest mean of error among different congestion models. Besides the circuit *adaptec*2, the results are very consistent. Although these two global routers apply two different approaches (AMGR mainly applied maze routing and MaizeRouter mainly applied Steiner tree routing and edge shifting), we can have similar results.

In figure 4.6, the congestion maps obtained by different congestion models

Table 4.7 Comparison on the mean of error of the congestion models when the circuit is global routed by AMGR

Test Cases	Lou's Model (μ)	Westra's Model (μ)	SMD Model (μ)	Detour Model (μ)	3-step Approach (μ)
*adaptec*1	22.55	21.28	20.78	21.77	19.88
*adaptec*2	18.77	22.09	22.50	23.26	22.37
*adaptec*3	8.98	5.54	4.74	5.02	4.23
*adaptec*4	6.82	3.79	3.49	3.54	3.31
*adaptec*5	21.06	13.64	12.07	12.94	9.68
*newblue*1	6.60	5.42	4.22	4.37	4.03
*newblue*2	5.53	3.75	3.08	3.09	3.02
Ave. (w.r.t. Lou's)	1.00	0.78	0.70	0.73	0.66

Table 4.8 Comparison on the mean of error of the congestion models when the circuit is global routed by MaizeRouter

Test Cases	Lou's Model (μ)	Westra's Model (μ)	SMD Model (μ)	Detour Model (μ)	3-step Approach (μ)
*adaptec*1	22.50	20.95	20.40	21.40	19.51
*adaptec*2	18.92	22.12	22.51	23.26	22.37
*adaptec*3	8.72	5.43	4.68	5.06	4.18
*adaptec*4	6.71	3.80	3.31	3.41	3.13
*adaptec*5	23.83	16.30	14.68	15.60	12.16
*newblue*1	6.67	5.16	4.21	4.32	4.03
*newblue*2	5.29	3.64	2.99	3.01	2.94
Ave. (w.r.t. Lou's)	1.00	0.78	0.71	0.73	0.67

and the actual one (obtained by global routing) are shown. We can see that there are many regions that are predicted as over-congested in the Lou's and Westra's models and there are also a lot of empty regions in their models. However, the nets can be ripped up and rerouted to avoid passing through the over-congested regions. There is thus no over-congested region after global routing and most of the tiles in the placed region are more or less used by some nets. In our modeling, we applied the preliminary estimation and congestion redistribution steps, and a similar congestion map can be obtained compared with the routing map. From these subjective results, we can conclude that our

model is more accurate than the other two.

4.8 Summary

In this chapter we analyze three models of congestion prediction in the early stages of VLSI physical design. SMD model is the fastest technique with the least amount of calculation involved. Detour model pays attention to surrounding tiles and 3-step approach with congestion redistribution is proved to be the most accurate one. Apparently, from the performance comparison we can conclude that our models have made significant improvement on accuracy of prediction compared to the previous models.

Chapter 5

Clock Network Synthesis

5.1 Overview

In this chapter, we propose two novel clock network synthesizers, DMST and DMSTSS. Some heuristics are proposed to optimize the CLR as well as the clock skew and the power usage. It mainly contains the following features: (1) A dual-MST based perfect matching algorithm is developed for symmetric clock tree construction. (2) A look-up table based hierarchical buffer sizing approach is developed for variation tolerance improvement. (3) An iterative buffer insertion technique is developed for delay balancing and capacitance reduction. (4) A dual-MZ blockage handling technique is developed for buffer location distribution and blockage avoidance. SPICE simulation is also applied for accurate delay estimation, the internal nodes of the clock tree are thus relocated for further skew minimization. A slew constraint ($\leq 100ps$) is employed in our synthesizer. We build a slew table for real-time reference during the execution of our program to satisfy this constraint.

In addition to the measurement of CLR, Monte Carlo simulation is also applied in the experiments. Monte Carlo simulation can simulate the variation of buffers, so as to provide a common method for performance evaluation of our clock network synthesizer. It can ensure that our proposed approach is practically variation-tolerant.

5.2 Problem Formulation

Let $T = \{V, E\}$ denote the clock tree. $V = \{v_i | i = 1, 2, \ldots, m_v\}$ is the set of nodes, and $E = \{e_j | j = 1, 2, \ldots, m_v - 1\}$ is the set of clock edges between the node v_j and its corresponding parent. Let $|e_j|$ denote the length of the edge e_j. We use $S = \{v_k | k = 1, 2, \ldots, m_s\}$ (where $m_s < m_v$) to denote the set of modules (the sinks, or leaf nodes). d_{s_i} denotes the signal delay of the ith sink. The rest $(m_v - m_s)$ nodes are named internal nodes. We use $|V|$, $|E|$ and $|S|$ to indicate the number of elements in V, E and S, respectively. The leaves are at level 0, and the root is at the highest level. Node v_i is said to be at level n_i if there are n_i edges on the path from v_i to the farthest leaf of the tree. Moreover, we assume that the topology of the clock tree is full binary, and every internal node has exactly two children. The skew of T is the difference between the longest signal delay and the shortest signal delay from the source to any sinks. Detailed definition can be found in the ISPD 2009 contest [88].

The restriction on clock slew describes the requirement on the signal transition time reduction. It is defined to be the rising time from 10% to 90% of the signal strength (90% to 10% for the falling time, respectively). The upper limit is set to be $100ps$. During the clock tree synthesis, it is necessary to maintain the signal transition time under this upper limit throughout the whole network.

CLR is the major metric in the performance evaluation. Two independent SPICE simulations under different voltage source (V_{dd_1} or V_{dd_2}) are applied. The value of CLR is determined by the difference between the maximal and minimal clock skew values under the two given voltage sources. Meanwhile, nominal clock skew is also considered, which is the maximal clock skew of all the voltage settings.

The power consumed by CMOS circuits consists of two components: static and dynamic power. The static power is mostly determined by the feature size

and other technology. Therefore, in this chapter we only consider dynamic power minimization. The definition of the dynamic power is $P = \frac{1}{2}\alpha C f V_{dd}^2$. C means the total load capacitance on the circuit, f is the frequency of the clock signal and V_{dd} is the power supply. α means the amount of switch times in each clock cycle. For clock tree $\alpha = 2$, because there is one rising and one falling edge in each clock period. Since α, f and V_{dd} are constant parameters in the digital circuits, we can use the total load capacitance C as a measure of the power usage. In the following sections, we try to minimize the capacitance for the power usage reduction. The upper limit of the total capacitance is predefined as an attribute of benchmarks to limit the power consumption.

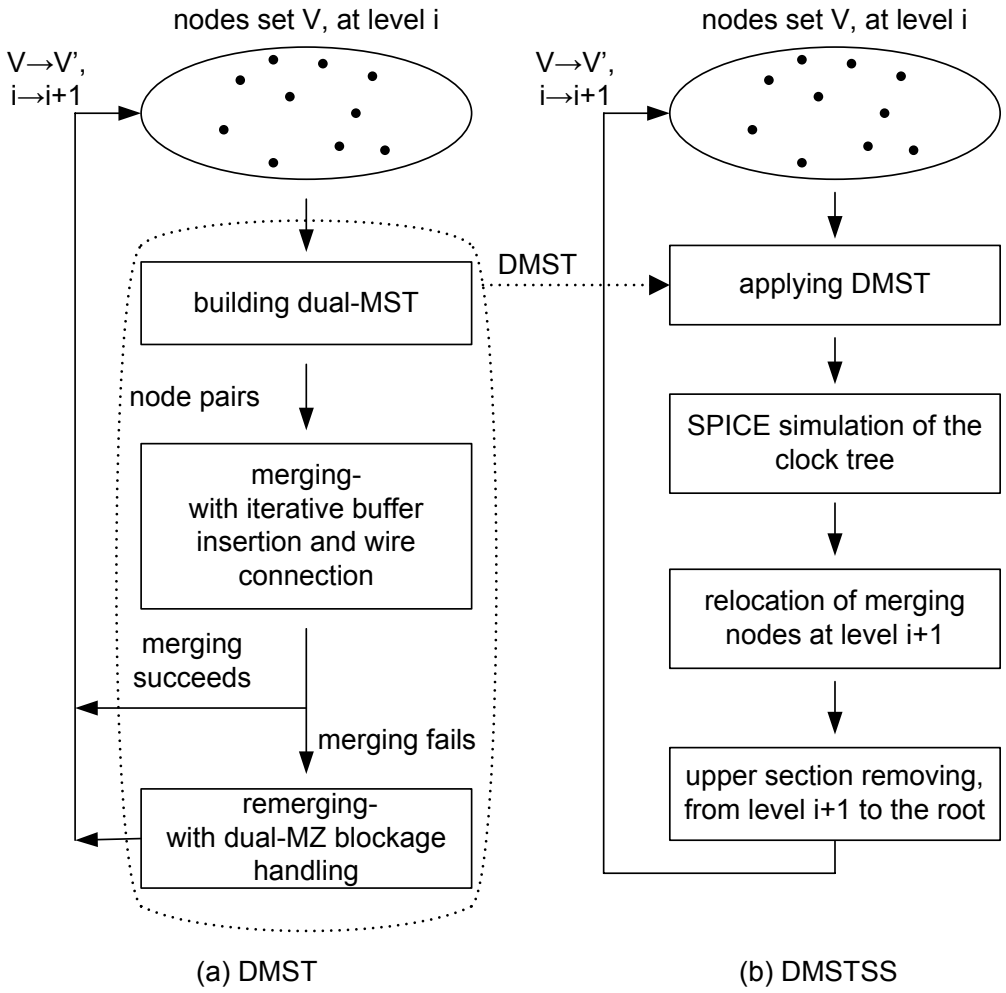
Given the physical location of all the modules (sinks), the objective of our work is to minimize the resultant CLR of each benchmark. Our work is subjected to the following three constraints:

1. Total capacitance cannot exceed the according limit of each benchmark.
2. Slew rate cannot exceed $100ps$ along the clock network.
3. Buffer cannot be placed on the blockage area.

5.3 Methodology

In this chapter, two novel clock network synthesizers, DMST and DMSTSS, are proposed for clock skew and load capacitance minimization. Both of the two synthesizers are designed in a bottom-up procedure with the application of DME [3, 10] technique. DMST is developed based on the construction of dual minimum spanning trees (dual-MST) in geometric matching. Elmore model computation is applied in DMST for delay balancing. DMSTSS is developed based on DMST with the inclusion of delay estimation from SPICE simulation and clock tree tuning.

Figure 5.1 Design flows of DMST and DMSTSS



DMST is an iterative approach, the general procedure of DMST is shown in figure 5.1(a). At the beginning of each iteration, we have a group of nodes to be merged. This is denoted as V. V can be composed of either clock sinks or internal nodes. A dual-MST based perfect matching technique is applied first to obtain a geometric matching solution upon the node group V. During the merging of each matched pair of nodes, a new technique of iterative buffer insertion and wire connection is applied to deal with the buffer distribution problem. If the merging fails because of blockages, this pair is re-merged with an exclusive dual-MZ blockage handling technique. After the successful merging of all the nodes in V and the generation of the nodes in V', this procedure is finished. It is then performed iteratively with V replaced by V', until a complete clock tree is constructed.

DMSTSS is a more advanced synthesizer. It is developed based on DMST with real-time SPICE simulation included for delay estimation. The flow of DMSTSS and its relationship with DMST are illustrated in figure 5.1(b). A clock tree is firstly built up by employing DMST. Then SPICE simulation is applied and the whole clock tree is specified with detailed signal delay information. Based on this, the internal nodes of each level are relocated for clock skew minimization. Notice that the synthesis result of DMSTSS is no longer a zero skew tree in terms of the Elmore model, but it has better performance in real circuits work. The details of our techniques are discussed in the following parts.

5.3.1 A Dual-MST based Geometric Perfect Matching

Traditional matching algorithms mainly focused on wirelength minimization [40, 24]. Our approach focuses on the construction of symmetric topology. In our clock tree, every root-to-leaf path will have similar buffer and wire distribution. This design will make the clock tree less sensitive towards process

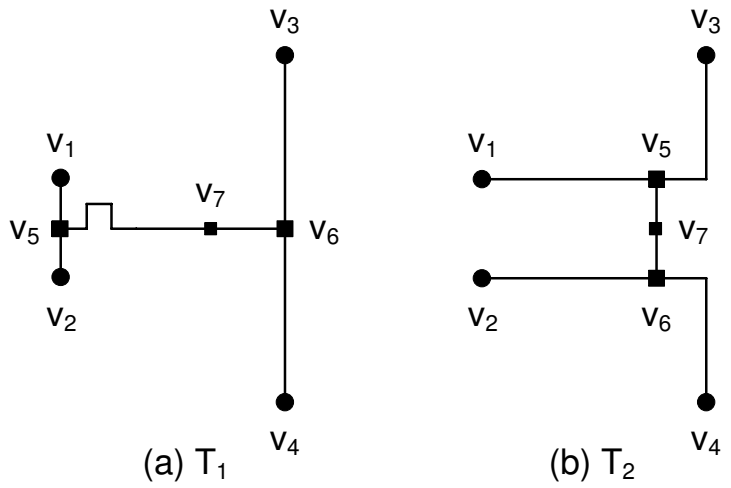
variation factors with the negative influence distributed in the sections of the clock tree more evenly. An example is shown in figure 5.2 to compare the differences between an asymmetric topology and a symmetric topology. In T_1, $V = \{v_1, \ldots v_4\}$ is a group of nodes. v_5 and v_6 are the internal nodes at the first level and v_7 is the root. The same naming rule is also applied in T_2. In both T_1 and T_2, all the root-to-leaf paths are equidistant, and the total wirelength of T_1 is shorter than that of T_2. However, T_2 is symmetric, but T_1 is asymmetric. In T_1 the branches in the same level differ, for instance, $dis\,(v_1, v_5) \neq dis\,(v_3, v_6)$. Therefore, the clock skew of T_1 is more sensitive towards process variation factors. T_2 is preferable in our work rather than T_1.

A dual-MST based geometric matching technique is developed for topology generation (a general definition of geometric matching can be found in [41]). It is a weighted perfect matching approach. Given a set of nodes $V = \{v_1, v_2 \ldots v_m\}$, we first construct a complete graph $G = \{V, E\}$. Let $|V|$ and $|E|$ denote the number of nodes and edges in the graph G, so$|V| = m$. Since G is a complete graph, every pair of two nodes v_i, v_j is connected by an edge $e_{i,j}$, $E = \{e_{1,2}, e_{1,3} \ldots e_{m-1,m}\}$ and $|E| = \frac{m(m-1)}{2}$. The cost of matching two nodes v_i and v_j is denoted as $f_c\,(e_{i,j})$. In our work we set $f_c(e_{i,j})$ to be the Manhattan distance between v_i and v_j. Let M denote the matching result of G. M is composed of a group of edges and it is a subset of E. The maximal pairing cost of M is denoted as C_{max} and defined as below. We will get close to a symmetric clock tree by reducing C_{max} in each level.

$$C_{max} = \max\{f_c\,(e_{i,j}) : \forall e_{i,j} \in M\} \tag{5.1}$$

At the beginning, the pairing cost f_c of every edge of E is computed, sorted in ascending order and stored in memory. Then the matching algorithm based on dual-MST is applied iteratively to generate every matching pair. The input of the approach is a complete graph $G = \{V, E\}$. Similar to the Kruskal's MST algorithm, edges in E are inserted orderly in terms of f_c. During this

Figure 5.2 Comparison of (a) an asymmetric tree and (b) a symmetric tree



Procedure 1 partition(G)

Require: $G = \{V, E\}$ is a complete graph, E is sorted in ascending order of $f_c(e_{i,j})$.
  if $|V| \leq 1$ **then**
    return;
  else if $|V| = 2$ **then**
    merge(v_1, v_2);
    return;
  else
    Building dual-MST with $|V| - 2$ edges inserted.
    Two subgraphs $G' = \{V', E'\}$ and $G'' = \{V'', E''\}$ are generated
    Two minimum spanning trees st' and st'' for V' and V'' are generated
    if $|V'|$ is odd and $|V''|$ is odd **then**
      $e_{m,n} = \arg_{e_{i,j}} \min\{f_c(e_{i,j}) \,|\forall n_i \in V', \forall n_j \in V''\}$;
      merge(v_m, v_n);
      remove v_m from V';
      remove v_n from V'';
      remove $e_{m,x}$ from E', $\forall x \in V'$;
      remove $e_{n,y}$ from E'', $\forall y \in V''$;
    end if
    partition(G');
    partition(G'');
    return;
  end if

procedure, no cycle is allowed. When $|V|-2$ edges have been inserted, two disjoint sub-trees st' and st'' are generated. Meanwhile, G is divided into two disjoint complete sub-graphs $G'=\{V',E'\}$ and $G''=\{V'',E''\}$. Notice that all the nodes of V' are connected by st', and all the nodes of V'' are connected by st'', respectively. It is obvious that st' and st'' are two minimum spanning trees for G' and G'', so dual-MST have been built up. If either $|V'|$ or $|V''|$ is an even number, it goes to the next iteration. Otherwise, an edge subset E_0 will be extracted from E. Every edge $e_{i,j}$ of E_0 has one node v_i from V' and the other node v_j from V''. The edge $e_{m,n}$ of E_0 with the minimum cost value f_c is determined as a matching pair.

$$f_c\left(e_{m,n}\right)=\min\{f_c\left(e_{i,j}\right):\forall e_{i,j}\in E_0\} \tag{5.2}$$

Notice that this dual-MST based matching approach is a geometric perfect matching. This means $2\times\lfloor\frac{|V|}{2}\rfloor$ nodes will be matched, and $\lceil\log_2|S|\rceil$ levels are performed. Assume that we are at the ith level with the node group V_i. The time complexity of f_c computation equals $O\left(|V_i|^2\right)$. On average, the number of partitioning stages is $\lceil\log_2\left(|V_i|\right)\rceil$. Thus the time complexity of partitioning is calculated as below.

$$O\left(\sum_{j=1}^{\lceil\log_2(|V_i|)\rceil}\left(\frac{1}{2}\times\frac{|V_i|}{2^{j-1}}\times\left(\frac{|V_i|}{2^{j-1}}-1\right)\times 2^{j-1}\right)\right) \tag{5.3}$$

Therefore, the total complexity of the ith level is still $O(|V_i|^2)$ on average. While $|V_i|=\lceil\frac{|S|}{2^{i-1}}\rceil$, the time complexity on average for the whole geometric matching is shown below.

$$O\left(\sum_{i=1}^{\lceil\log_2|S|\rceil}\lceil\frac{|S|}{2^{i-1}}\rceil^2\right)=O\left(|S|^2\right) \tag{5.4}$$

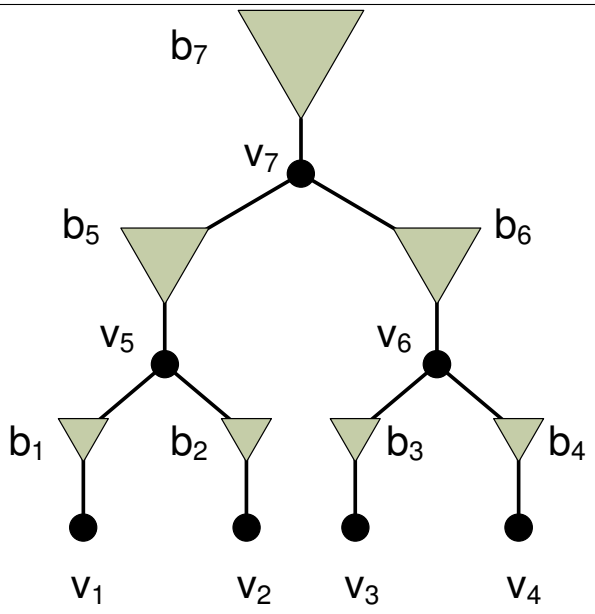
Our approach aims at constructing a symmetric clock tree but it is only a heuristic giving close proximity to the optimal solution. Compared to other

geometric matching methods, ours might generate longer wire length but would help to boost the performance of the clock tree. Specific comparison can be found in the section of the experimental results.

5.3.2 Hierarchical Buffer Sizing

The slew rate (transition time) of the clock signal should always be observed in order to guarantee the performance of a digital system. Thus, buffer insertion becomes a necessary step in clock tree synthesis. A clock buffer with larger size can provide larger driving power. It gives a smaller slew rate and less clock buffers is needed to be inserted. In addition, the signal delay of a larger buffer is less sensitive to variations. On the other hand, buffers with larger size will consume more power, and larger space is required for placement. Therefore, appropriate buffer sizing is an important issue in balancing the power consumption and variations.

Figure 5.3 Example of buffer sizing



The signal delay of a clock sink is determined by the path from the root to itself. In this chapter, it is named by a root-to-leaf signal path. The insertion

point at a higher level will occupy a larger number of such root-to-leaf signal paths. As shown in figure 5.3, v_7 at level 2 will cover 4 downstream clock sinks (v_1, v_2, v_3, v_4) and v_5 at level 1 will cover 2 downstream clock sinks (v_1, v_2). Therefore, we should enlarge the size of the buffers at higher levels to balance the signal delay of more clock sinks. Ideally, the clock buffers can be sized according to the example in figure 5.3. In this example, buffers with larger size will be inserted in upper levels and the smaller buffers will be used in lower level of the clock tree. In this chapter, an algorithm is devised to appropriately design the buffer size in descending order from the root to the leaves. Given one type of buffer, we connect a group of buffers in parallel to simulate larger buffer sizes.

Our main target in clock tree synthesis is to minimize the negative effect caused by variation factors. For this target, every root-to-leaf path should be constructed in similar pattern, with proper buffer location and sizing. Thus the number of buffer levels in each root-to-leaf path should be exactly the same. Moreover, the size of the buffer insertion in each buffer level should also be the same for all the sinks. Under this precondition, the size of the buffers at each insertion point is uniquely determined by its downstream buffer levels. It is a waste of time to devise the size for any specific insertion point during the synthesis. We develop a sizing rule based on some internal attributes of the clock sinks. During the procedure of clock tree synthesis, we compute the optimal size for each buffer level in advance and build up a table for buffer sizing reference. Subsequently, all the buffer insertion will follow this sizing table to determine its buffer size.

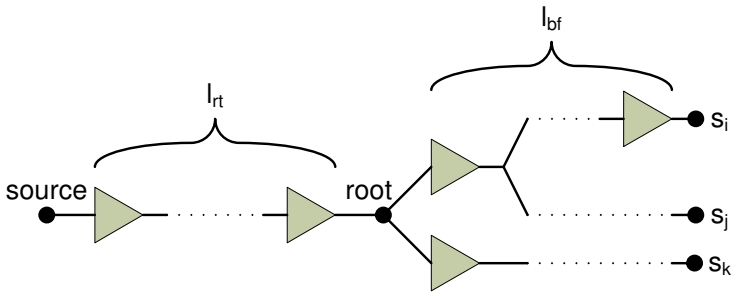
Here, we describe our rule for the construction of buffer sizing table. We define the total amount of different buffer sizes to be an integer m. Let $sz_b(x)$ denote the objective buffer size and x denote the amount of downstream buffer levels. According to figure 5.3, $sz_b(x)$ is a monotonically increasing function. Moreover, we define $L = \{L_0, L_1, \ldots, L_m\}$ to be the buffer levels between

different buffer sizes, as shown in figure 5.5. When $L_{i-1} \leq x < L_i$, $sz_b(x) = i$. L_0 is assumed to be zero, and L_m is assumed to be infinite. The rest of the elements in L is determined in the following way. An initial clock tree with the smallest buffer size (single buffer) at each insertion point is constructed first. In this initial tree, the number of buffer levels between the source node and the root is denoted as l_{rt}, and the number of buffer levels between the root to the clock sink within the longest root-to-leaf signal path is denoted as l_{bf}. An example is shown in figure 5.4. Based on the value of l_{rt} and l_{bf}, the buffer configuration is computed according to the following formula,

$$L_i = \lfloor (l_{bf} - l_{rt}) \times \frac{cap}{tot\_cap} \times \delta_i \rfloor \times 2 + 1 \tag{5.5}$$

where *cap* is the used capacitance of the initial clock tree, and *tot_cap* is the total capacitance allowed. δ_i is a parameter that is related to the driving power of i parallel buffers. Here i means the amount of parallel buffers and $i = 1, 2, \ldots, m-1$.

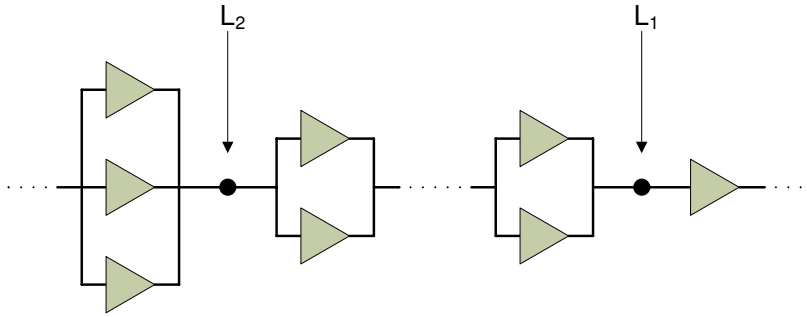
Figure 5.4 Definition of l_{rt} and l_{bf}



The size of buffers cannot be arbitrary in the real cases, and we connect a number of buffers in parallel to enlarge the size. Notice that we don't use node levels but downstream buffer levels to determine the buffer sizes. Let l_{v_i} denote the amount of downstream buffer levels at an internal node v_i. The amounts of buffer levels of its children nodes are denoted by $l^L_{v_i}$ and $l^R_{v_i}$. Hence, the value of l_{v_i} is determined by the following formula.

$$l_{v_i} = max\{l^L_{v_i}, l^R_{v_i}\} \tag{5.6}$$

Figure 5.5 A clock tree divided by buffer levels L_1 and L_2



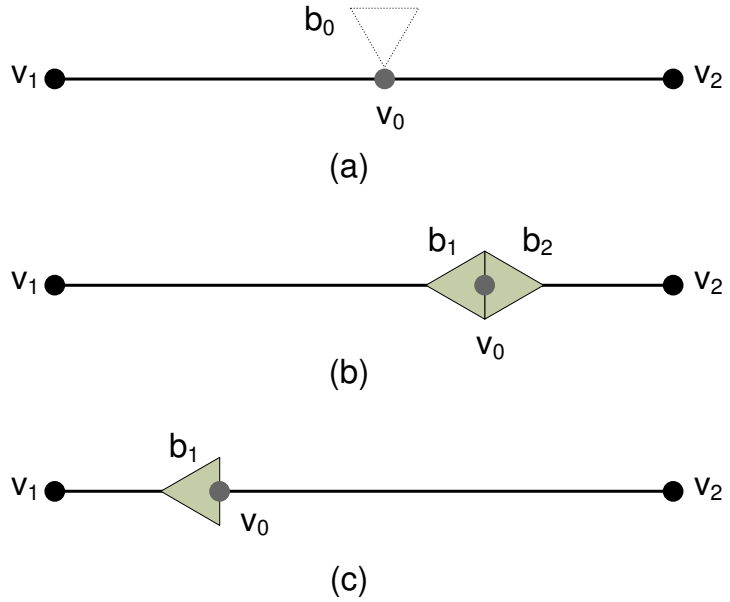
$l_{v_i} = 0$ if v_i is a sink node. Since it is a bottom-up method, the corresponding buffer levels of the internal nodes can be computed in this way.

5.3.3 Iterative Buffer Insertion

In our synthesizer of DMST, the topology is constructed level by level with concurrent buffer insertion and wire connection. This is mainly because accurate sub-tree information is important at geometric matching in each level. In DMSTSS, we will iteratively remove and rebuild the upper levels of the clock tree for merging point tuning. Global buffer distribution in the whole clock tree may lead to local cost increment for other tree sections. However, these benefited tree sections may be removed in the further tuning. Thus, it is improper to distribute the buffers globally along the whole tree. We developed a greedy algorithm of iterative buffer insertion, which is mainly focus on local optimization with skew balancing and load capacitance minimization.

An example is illustrated in figure 5.6 to demonstrate our approach for merging two nodes v_1 and v_2. Let $e_{1,2}$ denote the edge connecting the two nodes. Based on their downstream capacitances and sub-tree delays, the location of the delay balancing point, v_0, can be computed. Let l_{v_1} denote the amount of downstream buffer levels of v_1, and l_{v_2} denote the amount of downstream buffer levels of v_2, respectively. Generally, our method of buffer distribution can be divided into the following two parts.

Figure 5.6 Design flow of iterative buffer insertion



(1) $l_{v_1} = l_{v_2}$. We assume an upstream buffer b_0 at v_0, as shown in figure 5.6(a). If b_0 can drive v_1 and v_2 directly, the merging of v_1 and v_2 is finished. Otherwise, two back-to-back buffers b_1 and b_2 will be placed in a new delay balancing point along $e_{1,2}$, as shown in figure 5.6(b). The distance between b_1 and b_2 is set to be zero, and they are both located at v_0. Therefore, the current load capacitance left to the upper level is minimized. If no slew violation occurs, the merging is finished. Otherwise, assume there is slew violation along e_{v_1,b_1}. b_2 will be removed and b_1 will be shifted leftwards to its nearest non-slew-violation location. After the insertion of b_1, we enter the next iteration.

(2) $l_{v_1} \neq l_{v_2}$. Assume that $l_{v_1} \leq l_{v_2}$. We will insert a buffer b_1 along $e_{1,2}$, following the rule that b_1 is in the same location of v_0, as shown in figure 5.6(c). Therefore, the current load capacitance left to the upper level is minimized. If no slew violation occurs, we enter the next iteration with the insertion of b_1. Otherwise, there must be slew violation along e_{v_1,b_1}. b_1 will be shifted leftward to its nearest non-slew-violation point. After the insertion of b_1, we enter the

next iteration.

Notice that in the next iteration, v_0 replaces v_1 as an input node, and v_2 is remained. Similarly, $e_{0,2}$ replaces $e_{1,2}$. In our discussion of iterative buffer insertion, only straight wire connection is described in details. However, snaking cannot be completely avoided in some particular cases. When the delay difference between the two nodes cannot be solved with straight wire balancing, snaking wire technique can be applied.

5.3.4 Dual-MZ Blockage Handling Technique

Because of the existence of macro blocks, there may be pre-assigned obstacles on the chip area. Wires can cover these blockage areas freely, but buffers will cause violation. The connection is thus constrained. Traditionally, clock tree synthesizer can be divided into two steps. They do clock routing first to connect all the sinks to the signal source. Subsequently, buffer sizing, insertion and wire sizing are determined then. In case of buffer violation in the second step, clock routing has to detour out of the blockage areas in the first step. An example of possible result of traditional synthesis is shown in figure 5.8(a).

Instead of above mentioned approach, our approach considers concurrent buffer sizing and insertion during the procedure of clock routing. A novel blockage handling method, dual-MZ, is developed based on the traditional maze routing technique. Several enhancements are developed to make it satisfy the clock network synthesis constraints. We first decompose the chip into a $M \times N$ grid. Assume that v_1 and v_2 are the two nodes to be merged. Two independent maze routers mz_1 and mz_2 are initialized for v_1 and v_2, exclusively. Therefore, a dual-MZ is constructed. The priority queue of each maze router is sorted by the downstream delay values of its elements in ascending orders. Let dly_1 and dly_2 denote the minimum downstream delays of mz_1 and mz_2. Without loss of generality, assume that $dly_1 \leq dly_2$, and mz_1 is selected to be

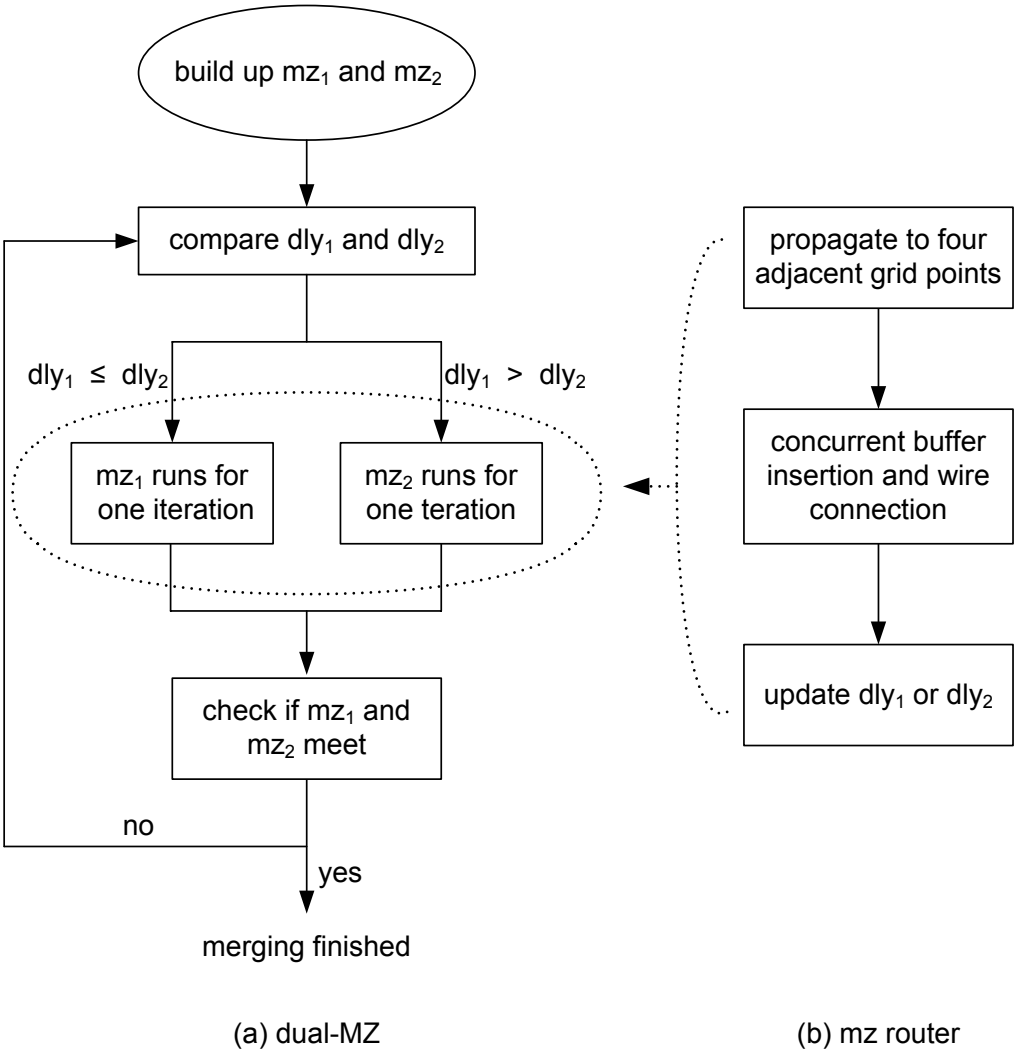
the primary router. Like ordinary maze routing, it will search the four adjacent grid points in one iteration. The downstream delay of mz_1 is then increased, so dly_1 will be updated. When dly_1 becomes bigger than dly_2, mz_2 will replace mz_1 to be the primary router and continue the routing job. In this way, mz_1 and mz_2 are executed in turns, and the values of dly_1 and dly_2 stay close to each other. A design flow of dual-MZ is shown in figure 5.7. When the two routers meet at the end, their accumulated delays will not differ a lot. dual-MZ can result in less buffer insertions and wire detours, so the power consumption is reduced. Notice that during this procedure, wire connection together with buffer insertion is concurrently applied along the propagation trace of the two routers, and they both contribute to the delay accumulation for comparison. A possible synthesis result of dual-MZ is shown in figure 5.8(b).

Our dual-MZ can save resources with less buffers and wires cost involved. Notice that the size of the grid graph ($M \times N$) is manually determined in our algorithm. M and N can be scaled in order to derive a tradeoff between routing complexity and routing quality. In our experiments, both M and N are set to be 1000. A synthesis result of the benchmark $ispd09fnb1$ is shown in figure 5.9 (the detailed information of $ispd09fnb1$ can be found in table 5.2). The green stars denote the clock sinks, the blue squares denote the buffers, the purple circle denotes the signal source, the red lines denote the wire connection and the gray rectangles denote the blockage areas. We can see that by applying dual-MZ, clock wire can cover the blockage area freely with proper buffer insertion outside. Resources cost can be reduced by our method efficiently.

5.3.5 Merging Point Relocation with SPICE Simulation

DMSTSS is an advanced version of DMST with SPICE tuning involved. Given a clock tree, SPICE simulation can provide accurate delay estimation at any

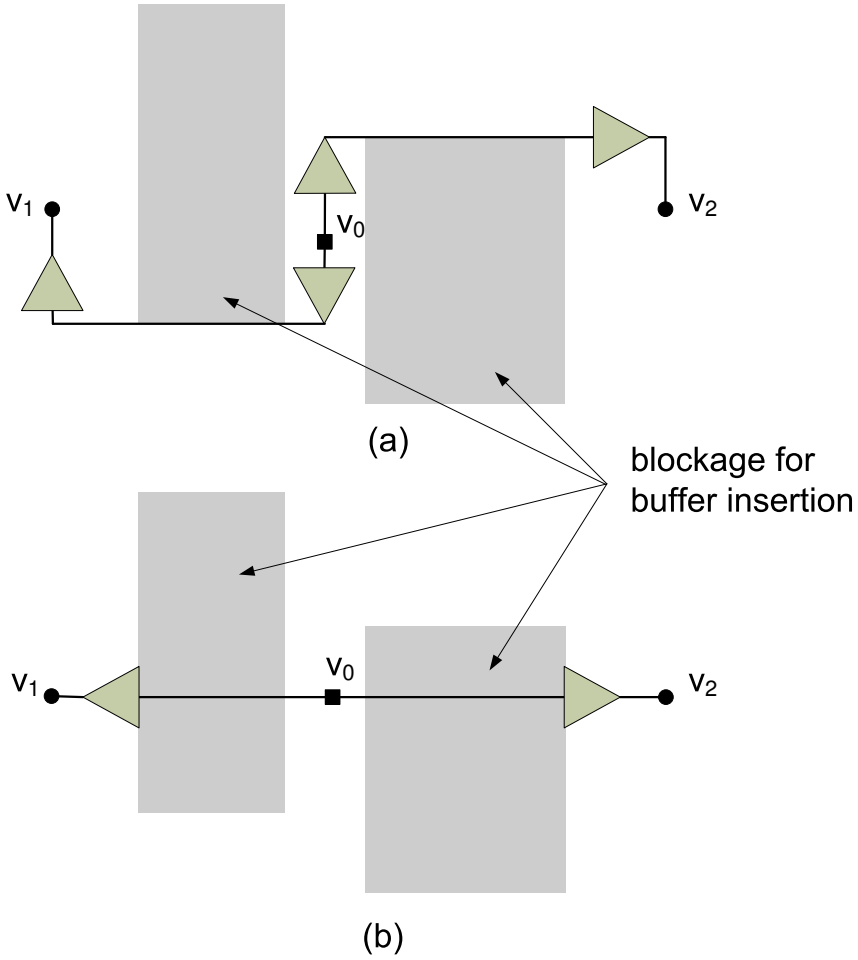
Figure 5.7 Design flow of (a) dual-MZ and (b) specific maze routing



point. Based on this additional information, clock tree can be adjusted to improve the deviation caused by Elmore model. Nevertheless, despite improvement on clock skew, the runtime cost of SPICE simulation is quite large. Thus we need to reach a tradeoff between performance and efficiency.

Our merging point relocation is a level-by-level approach. Here the level means not buffer level but the node level in the tree structure. The specific flow is shown in figure 5.1(b). After DMST, a clock tree is available. SPICE simulation of the tree is applied to get the delay estimation. Based on this

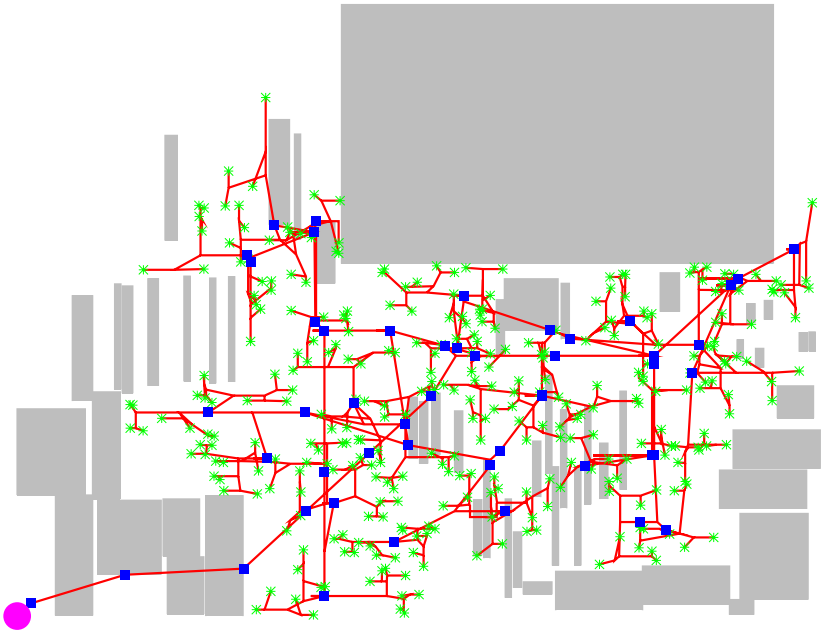
Figure 5.8 Possible synthesis results of (a) complete detour and (b) dual-MZ



accurate estimation, all the internal nodes in the current level will be relocated simultaneously. Then the tree sections at upper levels are removed and rebuilt by DMST, based on the set of nodes at the current level. Because of this iterative removing and rebuilding, slew violation caused by merging point relocation can be avoided. Meanwhile, the upper topology of the tree is dynamically improved. DMSTSS is applied in this iterative way, it ends when the root is reached.

The relocation of each merging point is based on Elmore delay computation of wire connections. No buffer computation is involved in this tuning approach. Assume that v_0 is the node to be relocated, and v_1 and v_2 are its two children

Figure 5.9 DMSTSS synthesis result of the benchmark $ispd09fnb1$



nodes. $e_{0,1}$ and $e_{0,2}$ are the two according edges. We first update the delay values of the nodes with SPICE estimation result. Then a bidirectional search is applied from v_0 to v_1 and v_2, along $e_{0,1}$ and $e_{0,2}$. This search stops when it meets the first buffer on the edge or it meets the ending node. Assume the two stops of the bidirectional search to be n_1 and n_2 on $e_{0,1}$ and $e_{0,2}$, respectively. Then the two wire connections e_{v_0,n_1} and e_{v_0,n_2} are removed and remerged. Notice that here Elmore model is still used in delay computation, but the specific delay values are from SPICE estimation.

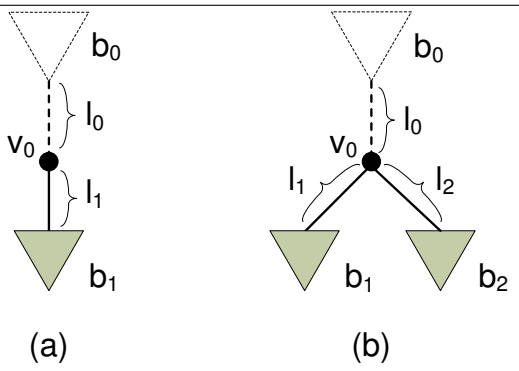
5.3.6 Slew Table Construction

In our work, slew rate constraint ($\leq 100ps$) is engaged along the whole clock tree. Real-time slew tuning will cost a lot of time and decrease the efficiency of the program. We pre-build a look up table for slew reference during the execution of our program. The procedure to construct the slew table can be divided into two categories, single wire and binary branch. At any node, we

need to get the maximum driving length from the slew table, with upstream buffer size, downstream buffer size and downstream wirelength as the indexes. This is shown in figure 5.10. We can assume that the current node to be v_0, and l_0 is the maximum driving length for the next buffer insertion b_0. l_0 is what we need to get from the slew table in the program. As shown in figure 5.5, the buffer size sz_{b_0} for the next level can be derived from our buffer sizing table, so it is a know value. At the single wire in figure 5.10(a), downstream information of sz_{b_1} and l_1 are ready to utilize. Therefore, with three available indexes of sz_{b_0}, sz_{b_1} and l_1, we can get l_0 from the slew table. We can get l_0 for a binary branch in the same way, as shown in figure 5.10(b). The only difference is that there are two additional indexes required for the slew table, which are sz_{b_2} and l_2 as the downstream buffer size and wire length of the other branch.

As stated in 5.3.2, buffer size are defined by the number of parallel buffer connection in our work. Thus sz_{b_0}, sz_{b_1} and sz_{b_2} are discrete values (integers actually) and they can be used as table indexes directly. However, wire length is a real continuous value in nanometer scale. Therefore, we quantize the wire length of l_0, l_1 and l_2 during the slew table construction in order to maintain the table size within a proper range. During the clock tree synthesis, a linear interpolation program is applied to transfer the actual wire length into discrete table index.

Figure 5.10 Driving length reference at (a) single wire and (b) binary branch



5.4 Experimental Results

In this section, our experimental results are presented. We implement our clock tree synthesizer in the C language and the program is executed on the Linux operating system with an Intel Core2 Quad 2.4GHz CPU and 4GB memory. In our experiment, one type of wire and one type of buffer is used in our clock tree synthesizer. The unit resistance of the wire is $0.0003\Omega/nm$, and the unit capacitance of the wire is $0.00016fF/nm$. The specific configuration of the buffer in different sizes is shown in table 5.1. In our synthesizer, the maximum buffer size is set to be 6, which is inserted on the connection path between the source and the root. Hence we list the attributes of different buffer sizes less than or equal to 6. This table is generated from our SPICE simulation statistics. C_b means the input capacitance, R_b means the driver resistance and d_b means the internal delay of a buffer, respectively. During the evaluation, two power supplies with $V_{dd_1} = 1.0V$ and $V_{dd_2} = 1.2V$ are employed independently for the simulation of voltage variation and CLR computation.

Table 5.1 Buffer configuration

buffer sizes	$C_b\,(fF)$	$R_b(\Omega)$	$d_b(ps)$
1	35	66.9	4.92
2	70	40.5	5.63
3	105	31.3	6.13
4	140	26.4	6.52
5	175	25.0	6.95
6	210	20.7	7.20

Two benchmarks suites are included in our experiment. The first one is released from the ISPD 2009 contest [88]. Detailed information of the benchmark circuits is shown in table 5.2.

There are three traditionally common geometric matching methods, which are MMM [40], KCR [41] and CL [24]. MMM applies a top-down approach to

Table 5.2 Circuit information of the benchmarks from ISPD 2009

Circuits	Chip Size (mm x mm)	No. of sinks	No. of block (Area %)	limit CAP (fF)
ispd09f11	11.0 x 11.0	121	0 (0%)	118000
ispd09f12	8.1 x 12.6	117	0 (0%)	110000
ispd09f21	12.6 x 11.7	117	0 (0%)	125000
ispd09f22	11.7 x 4.9	91	0 (0%)	80000
ispd09f31	17.1 x 17.1	273	88 (24.38%)	250000
ispd09f32	17.0 x 17.0	190	99 (34.26%)	190000
ispd09fnb1	2.6 x 2.1	330	53 (37.69%)	42000
ispd09f33	15.3 x 15.3	209	80 (27.68%)	195000
ispd09f34	16.0 x 16.0	157	99 (38.67%)	160000
ispd09f35	15.3 x 15.3	193	96 (33.22%)	185000
ispd09fnb2	6.4 x 4.4	440	1346 (63.88%)	88000
avg.	12.1 x 11.6	203	169 (23.62%)	140273

construct the clock tree, and it cannot be embedded into our clock network synthesizer. We apply a bottom-up approach, which is based on the geometric matching procedure of KCR [41]. Other than the traditional bottom-up merging procedure of KCR, dual-MST is newly developed in our perfect matching. For additional performance comparison, we implement CL [24] in our synthesizer to replace the dual-MST based matching method, with the other components of our synthesizer unchanged. In CL, the cluster size is maintained to be $\frac{2}{3}$ of the group of nodes (in [24], the ratio is between $\left(\frac{1}{2}, 1\right)$ so $\frac{2}{3}$ is a proper setting). The result of comparisons is shown in table 5.4. The result is obtained from execution on the benchmarks in table 5.2, and SPICE simulation is involved during clock tree synthesis for each case. In the performance evaluation, Monte Carlo simulation is performed based on the variations of the buffers. The main parameters of the transistors with variations to construct the buffer are shown in table 5.3.

In table 5.4, nominal clock skew (ps) (average value obtained after Monte

Table 5.3 Variation of the transistor

Parameters	NMOS			PMOS		
	Typical	Slow	Fast	Typical	Slow	Fast
$l_{int}(10^{-9})$	3.750	2.875	4.625	3.75	2.875	4.625
v_{th0}	0.404	0.431	0.377	-0.384	-0.411	-0.355
k_1	0.400	0.420	0.379	0.400	0.422	0.378
u_0	0.054	0.051	0.057	0.020	0.018	0.022
$x_j(10^{-8})$	1.40	1.54	1.26	1.40	1.54	1.26

Carlo simulation with random variations of the buffers), capacitance percentage ($C\%$) and CPU time (seconds) are shown for performance comparison. Compared to the approach of CL, the nominal clock skew obtained by dual-MST approach is similar in some cases and the average nominal clock skew can be reduced significantly in some particular cases. It is because the cost difference between the matching pair with the smallest cost and the matching pair with the largest cost are larger while CL is applied. This results in more snaking wires for skew balancing and hence more buffers should be inserted. The clock tree becomes less variation-tolerant when there are more buffer insertions. In addition, buffers cannot be inserted at some blockage regions. The total capacitance can be reduced significantly and so as the power consumption while there are less buffer insertions. It is also the reason that our matching technique, dual-MST, can outperform CL in total capacitance and CPU time.

A summary of the performance of different clock tree synthesizers is shown in table 5.6, table 5.7 and table 5.8. CLR (ps), clock skew (ps), capacitance percentage (%) and CPU runtime (seconds) are listed for performance comparison. DMSTSS is our proposal in this chapter. As discussed in section 5.3, it is an extended work of DMST applying real-time SPICE simulation for clock tree tuning. We compare our results with three clock network synthesizers published in ASP-DAC 2010 (HKPU [55], NTU [86] and NCTU [53]) and the best result with the smallest CLR of each circuit published in the ISPD 2009

Table 5.4 Comparison between different matching methods

Circuits	CL			Dual-MST		
	Nominal skew	C%	CPU	Nominal skew	C%	CPU
ispd09f11	17.66	83.4	388	18.20	79.3	180
ispd09f12	17.37	91.9	446	17.50	89.3	213
ispd09f21	19.18	83.3	427	18.76	83.2	210
ispd09f22	16.28	91.7	288	14.00	79.4	113
ispd09f31	24.10	78.4	1182	24.46	83.4	777
ispd09f32	21.49	83.1	853	21.54	82.4	420
ispd09fnb1	13.65	152.3	346	12.58	82.0	82
ispd09f33	23.51	79.2	785	23.56	83.6	483
ispd09f34	20.02	83.2	647	20.59	85.7	354
ispd09f35	23.73	89.1	1190	22.19	85.0	453
ispd09fnb2	19.49	123.2	593	15.35	87.9	202
avg.	19.68	94.4	650	18.97	83.8	317

contest [88] (ISPD09). The computing platforms for the above synthesizers are shown in table 5.5.

Table 5.5 Comparison of computing platforms

Synthesizer	CPU	MEM
DMSTSS,HKPU	Intel Core2 2.4 GHz	4 GB
NTU	Intel Xeon 2.0 GHz	16 GB
NCTU	Intel Xeon 3.0 GHz	32 GB
ISPD09	AMD Opteron 2.8 GHz	128 GB

According to the results in table 5.6, table 5.7 and table 5.8, we can see that the CLR obtained from DMSTSS is outstanding. Generally, the average CLR of DMSTSS is only 81.5% of HKPU (86.2% in all 11 circuits), 47.7% of NTU, 54.6% of NCTU and 41.8% of ISPD09. Meanwhile, the average CPU time of our approach is only 107% of HKPU (108% in all 11 circuits), 5.4% of NTU, 10.8% of NCTU and 1.8% of ISPD09. CLR, as a combined evaluation index of both clock skew and voltage variation, is the first criterion in comparison. It

Table 5.6 Comparison among DMSTSS, the three synthesizers in ASP-DAC 2010 and the best result in ISPD 2009.

Circuits	DMSTSS				HKPU [55]			
	CLR	SKEW	C%	CPU	CLR	SKEW	C%	CPU
ispd09f11	10.1	5.5	87.4	208	12.2	6.5	79.3	180
ispd09f12	8.8	5.4	93.2	227	10.9	7.0	89.3	213
ispd09f21	9.7	6.3	93.9	258	12.1	6.7	83.2	210
ispd09f22	6.9	4.2	89.2	136	9.9	4.3	79.4	113
ispd09f31	11.0	8.0	83.4	727	13.4	10.1	83.4	777
ispd09f32	9.9	7.6	85.2	485	11.5	8.8	82.4	420
ispd09fnb1	12.1	6.2	91.6	95	13.8	7.8	82.0	82
avg.(7)	9.8	6.2	89.8	305	12.0	7.3	82.7	285
ispd09f33	13.6	10.2	88.8	531	13.4	7.9	83.6	483
ispd09f34	9.9	8.5	91.0	385	11.3	6.2	85.7	354
ispd09f35	11.3	9.9	90.8	507	13.1	11.1	85.0	453
ispd09fnb2	13.1	8.7	87.9	202	13.1	8.7	87.9	202
avg.(11)	10.6	7.3	89.7	342	12.3	7.7	83.8	317

can be seen that our DMSTSS has great advantage on CLR compared to NTU, NCTU and ISPD09. Even compared with our previous work HKPU, there is still more than 10% improvement. Meanwhile, our synthesizer DMSTSS is much faster than the proposed works. It is a little slower than that of HKPU with less than 10% additional runtime, but the total efficiency is still good. Additionally, due to the practicality problems of CLR (voltage supplies at different points will not vary simultaneously), we listed the nominal clock skew of each work as the SKEW (ps) in the tables. Notice that the clock skew of DMSTSS is improved based on HKPU, although it is still larger than the results of NTU, NCTU and ISPD09.

A summary of the performance of our clock tree synthesizer with fixed buffer sizes is shown in table 5.9 and table 5.10. No buffer sizing technique is applied here. CLR, capacitance percentage and CPU time are listed. N_{Bf} is the number of buffers used in every buffer insertion along the whole clock tree. We run our program with two scenarios, DMST and DMSTSS. With smaller

Table 5.7 Comparison among DMSTSS, the three synthesizers in ASP-DAC 2010 and the best result in ISPD 2009.

Circuits	NTU [86]				NCTU [53]			
	CLR	SKEW	C%	CPU	CLR	SKEW	C%	CPU
ispd09f11	19.7	4.5	NA	4639	18.8	7.1	NA	2200
ispd09f12	17.5	4.1	NA	4231	15.5	3.1	NA	1923
ispd09f21	19.9	3.9	NA	4629	17.0	3.0	NA	2231
ispd09f22	16.5	3.7	NA	3937	16.3	4.1	NA	1370
ispd09f31	31.1	4.8	NA	11112	22.6	7.6	NA	6360
ispd09f32	23.0	4.2	NA	7293	20.6	5.5	NA	3967
ispd09fnb1	15.7	6.8	NA	3719	14.3	3.8	NA	1743
avg.(7)	20.5	4.6	NA	5651	17.9	4.9	NA	2828
ispd09f33	NA	NA	NA	NA	NA	NA	NA	NA
ispd09f34	NA	NA	NA	NA	NA	NA	NA	NA
ispd09f35	NA	NA	NA	NA	NA	NA	NA	NA
ispd09fnb2	NA	NA	NA	NA	NA	NA	NA	NA
avg.(11)	NA	NA	NA	NA	NA	NA	NA	NA

buffer size, the CLR is larger and less capacitance is used. With larger buffer size, the CLR is smaller and the total capacitance is increased significantly. We can observe that buffer insertion with fixed size is not a good idea in clock tree synthesis, and there is a tradeoff between load capacitance and CLR. Compared to the results to DMSTSS shown in table 5.6, we can see that the capacitance can be utilized more efficiently with significant CLR reduction by using our buffer sizing technique discussed in 5.3.2.

Our DMSTSS synthesizer has good performance on the ISPD 2009 benchmarks shown in table 5.2. However, the average amount of clock sinks is only 203. Even the largest size of the benchmarks has only 440 sinks. These benchmarks are too small and they are less representative of the modern technologies. We choose five additional standard benchmarks, r1 to r5 from [92], to further test our synthesizers. The information of the benchmarks is shown in table 5.11. The average sink amount of r1 to r5 is 1346. Since there is no

Table 5.8 Comparison among DMSTSS, the three synthesizers in ASP-DAC 2010 and the best result in ISPD 2009.

Circuits	ISPD09 [88]			
	CLR	SKEW	C%	CPU
ispd09f11	22.3	6.3	89.9	23358
ispd09f12	22.2	5.4	87.9	14992
ispd09f21	19.6	3.2	86.7	26420
ispd09f22	16.4	3.0	85.0	9432
ispd09f31	45.1	7.6	73.5	40088
ispd09f32	18.4	7.7	89.9	2888
ispd09fnb1	19.8	7.2	63.1	477
avg. (7)	23.4	5.8	82.3	16807
ispd09f33	NA	NA	NA	NA
ispd09f34	NA	NA	NA	NA
ispd09f35	NA	NA	NA	NA
ispd09fnb2	NA	NA	NA	NA
avg. (11)	NA	NA	NA	NA

capacitance limit in the original benchmark attributes, we manually set an appropriate value of capacitance limit for each case. The performance statistics of DMSTSS for r1 to r5 are shown in table 5.12, with CLR (ps), SKEW (ps), CAP (fF) and CPU (seconds) listed. Notice that SKEW means the maximal clock skew value of the evaluation results under two voltage supplies and two signal signs. We can see that our synthesizer has good performance on this benchmark series. The average CLR and capacitance cost are 13.9 ps and 408765 fF on r1 to r5, respectively.

5.5 Summary

In summary, two clock tree synthesizers, DMST and DMSTSS, have been proposed. Several novel methods, including dual-MST perfect matching, buffer insertion and sizing, dual-MZ blockage handling, clock tree tuning with SPICE

Table 5.9 Comparison of CLR for fixed buffer sizing in DMSTSS

Circuits	DMSTSS					
	$N_{Bf} = 1$			$N_{Bf} = 3$		
	CLR	C%	CPU	CLR	C%	CPU
ispd09f11	57.6	49.9	73	23.46	105.1	320
ispd09f12	51.4	47.8	64	17.78	98.8	279
ispd09f21	58.1	48.0	76	20.59	105.9	341
ispd09f22	40.8	46.9	43	15.83	96.4	168
ispd09f31	81.0	50.7	295	24.26	107.6	1545
ispd09f32	80.3	51.1	163	23.71	110.9	910
ispd09fnb1	19.3	52.3	38	14.79	109.8	135
ispd09f33	73.6	50.8	225	21.36	102.5	946
ispd09f34	74.3	51.2	128	21.39	105.6	606
ispd09f35	84.5	50.1	236	24.86	109.6	1000
ispd09fnb2	32.1	54.2	95	16.53	123.5	355
avg.(11)	59.4	50.3	130	20.41	106.8	601

simulation and slew table construction are developed to solve the clock network synthesis problem. Our dual-MST based geometric matching method can construct a clock tree of symmetric structure with increased tolerance towards process variation. Our buffer sizing technique is able to utilize the capacitance more efficiently in order to reduce the negative effect of variations. Besides, iterative buffer insertion and dual-MZ blockage handling can solve the problem on connection of a merging pair with delay balancing and saving on resources usage. Internal nodes relocation further tunes the clock tree and reduces the clock skew. Meanwhile the slew table provides driving length of each buffer insertion during the program execution, in order to satisfy the slew rate constraint. Experimental results show that our improved synthesizer has good performance and high efficiency. From the comparison in table 5.6, table 5.7 and table 5.8 with the previous works based on the ISPD 2009 benchmark suite, we can see our synthesizer with the best performance on CLR and runtime. Additionally, the results in table 5.12 on the benchmarks r1 to r5 shows consistently good performance thus robustness of our DMSTSS synthesizer

Table 5.10 Comparison of CLR for fixed buffer sizing in DMST

Circuits	DMST					
	$N_{Bf} = 1$			$N_{Bf} = 3$		
	CLR	C%	CPU	CLR	C%	CPU
ispd09f11	62.2	49.1	0.6	31.7	103.3	0.6
ispd09f12	60.2	48.5	0.6	31.1	101.3	0.6
ispd09f21	71.2	51.3	0.6	30.0	108.0	0.7
ispd09f22	50.3	45.5	0.6	27.9	91.8	0.6
ispd09f31	94.0	51.7	0.9	37.3	108.8	1.6
ispd09f32	88.9	51.1	0.7	34.4	114.9	0.8
ispd09fnb1	31.6	50.0	1.1	39.9	115.4	1.1
ispd09f33	84.6	51.5	0.8	35.2	104.1	1.8
ispd09f34	79.3	50.8	0.7	34.0	108.0	0.7
ispd09f35	95.0	51.0	2.9	58.9	108.8	3.1
ispd09fnb2	46.4	53.2	2.2	17.1	123.3	2.2
avg.(11)	69.4	50.7	1.1	35.1	108.0	1.3

on larger circuits. The result from Monte Carlo simulation shows that our topology is more tolerant on the process variations.

Table 5.11 Circuit information of the benchmarks from r1 to r5

Circuits	Chip Size (mm x mm)	No. of sinks	No. of block (Area %)	limit CAP (fF)
r1	8.6 x 8.6	267	0 (0%)	170000
r2	11.6 x 11.7	598	0 (0%)	330000
r3	12.2 x 12.0	862	0 (0%)	450000
r4	15.9 x 15.9	1903	0 (0%)	950000
r5	18.2 x 17.8	3101	0 (0%)	1500000
avg.	13.3 x 13.2	1346	0 (0%)	680000

Table 5.12 Performance of DMSTSS on r1 to r5

Circuits	DMSTSS			
	CLR (ps)	SKEW (ps)	CAP (fF)	CPU (s)
r1	12.8	9.2	129200	391
r2	11.0	8.3	222514	819
r3	13.3	11.1	286208	1150
r4	13.0	11.4	561388	4266
r5	19.6	18.9	844516	9407
avg.	13.9	11.8	408765	3207

Chapter 6

Clock Gating Design

6.1 Overview

In this chapter, we propose two novel synthesizers, HKPUst and HKPUcg, to construct a binary clock tree in a bottom-up course. Simultaneous optimization on the clock skew and the power dissipation is applied. The topology generator is responsible for a buffered and gated clock tree, and the clock gates are inserted concurrently. In our work, the downstream masking information of subtrees is taken into account during each merging step. Two algorithms of topology generation, dual-MST [55] and NNS [24], are applied in our work. The according cost functions are improved for power awareness, therefore it can reduce the switched capacitance. HKPUcg applies the same slew rate ($C_d \leq 20 \times C_g$) as previous works [65, 11]. Besides, in HKPUst we perform a more strict slew constraint ($\leq 100ps$) along the whole clock network. Thus the constraint on buffer and gate location is emphasized. The experimental results show that our method can greatly reduce the power consumption of the clock network with proper gate insertion. Meanwhile, the nominal clock skew is guaranteed to be exact zero. With the simulation of SPICE, the resultant skew is still acceptable.

6.2 Problem Formulation

6.2.1 Clock Tree and Controller Tree

Let $T = \{V, E\}$ denote the clock tree. $V = \{v_i | i = 1, 2, \ldots, m_v\}$ is the set of nodes, and $E = \{e_j | j = 1, 2, \ldots, m_v - 1\}$ is the set of clock edges between the node v_j and its corresponding parent. Let $|e_j|$ denote the length of the edge e_j. Apparently, for the root node there will be no edge assigned. Let $G = \{g_i | i = 1, 2, \ldots, m_v - 1\}$ denote the set of gates. The gate g_j is assigned to be on the edge e_j masking the node v_j directly. We use $S = \{v_k | k = 1, 2, \ldots, m_s\}$ (where $m_s < m_v$) to denote the set of modules (the sinks, or leaf nodes). The rest $(m_v - m_s)$ nodes are named internal nodes. The root is said to be at level 0. Node v_i is said to be at level n_i if there are n_i edges on the path from v_i to the root of the tree. Moreover, we assume that the topology of the clock tree is full binary. Every internal node has exactly two children. The skew of T is the difference between the longest signal delay and the shortest signal delay from the source to any sinks. As proposed in [65], we assume that the control logic is located at the center of the chip. Star routing is also applied in the controller tree, denoted as T^{ctr}. A control edge EN_i in T^{ctr} will transmit the enable signal to the respective gate g_i on the edge e_i in the clock tree T. An example of a clock tree T as well as its controller tree T^{ctr} is shown in figure 6.1.

During the operating time of a circuit, each module will have its active and idle times. It is usually specified as different activity patterns. The activity patterns can be obtained by the simulation of the design at the behavioral level [27]. Let A_i denote the activity pattern of the node v_i. It is a binary string with 1s indicating the active periods and 0s indicating the idle periods of a sink or an internal node. If v_i is a sink node, we can directly obtain A_i from the benchmark file. Otherwise, suppose v_i to be an internal node with two children nodes v_L and v_R accordingly. The clock signal at v_i must be

enabled whenever its left or right child is active. Therefore, A_i is calculated by performing the bitwise OR operation on the activity patterns of v_L and v_R. Hence $A_i = A_L \cup A_R$. An example of a bottom-up activity pattern transmission is shown in figure 6.2.

Let $P(A_i)$ denote the activity of the node v_i, and $P_{tr}(A_i)$ denote its transition probability. These two factors are calculated based on the corresponding pattern of v_i. The specific equations are shown as below

$$P(A_i) = \frac{AT_{no}(A_i)}{Len(A_i)}, P_{tr}(A_i) = \frac{TR_{no}(A_i)}{2 \times (Len(A_i) - 1)} \tag{6.1}$$

where $AT_{no}(A_i)$ is the number of active times (1s) in A_i, and $TR_{no}(A_i)$ is the number of transitions (10 or 01) in A_i. $Len(A_i)$ denotes the stream length of A_i.

Figure 6.1 A gated clock binary tree.

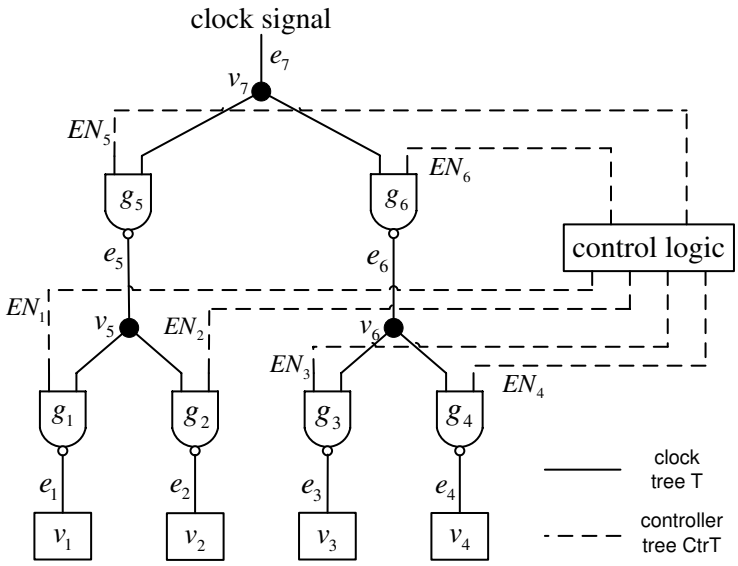
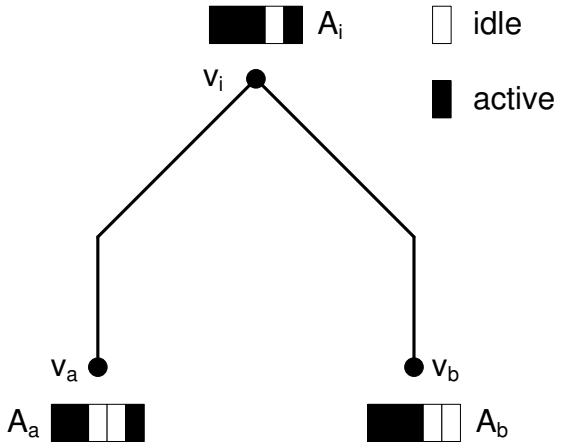


Figure 6.2 An example of activity pattern transmission.



6.2.2 Switched Capacitance

The power consumed by CMOS circuits consists of two components: static and dynamic power. The static power is mostly determined by the feature size and other technology. In this chapter, we only consider dynamic power minimization. The definition of the dynamic power is $P = \frac{1}{2}\alpha C f V_{dd}^2$. C means the total load capacitance on the circuit, f is the frequency of the clock signal and V_{dd} is the power supply. α means the amount of switch times in each clock cycle. For clock tree $\alpha = 2$, because there is one rising and one falling edge in each clock period. $\alpha = 1$ in the controller tree, respectively. Since f and V_{dd} are constant parameters in the digital circuits, we can use the switched capacitance as a measure of the power usage. Assume that a subtree T_i rooted at v_i with a gate insertion g_i, and the controller tree is denoted as T_i^{ctr}. The unmasked load capacitance for T_i and T_i^{ctr} are $C_{v_i}^u$ and $C_{T_i^{ctr}}^u = C_{EN_i} + C_g$ accordingly, C_g denotes the input capacitance of a gate. We can get the equation for the downstream switched capacitance of v_i as $SC_{v_i} = C_{v_i}^u P(A_i)$. Similarly, the corresponding switched capacitance for the controller tree T_i^{ctr} is measured as $SC_{T_i^{ctr}} = (C_{EN_i} + C_g) P_{tr}(A_i)$.

The power consumption of a clock network is directly proportional to the

average switched capacitance for each clock cycle. The total switched capacitance is contributed by a gated and buffered clock tree T and a controller tree T^{ctr}. In order to reduce the switching activity, modules and clock tree sections can be disabled by clock gates during their inactive clock periods. From the above example, we can see that the original capacitance of node v_i is $C^u_{v_i}$. With gate g_i inserted at v_i, the resultant switched capacitance is $SC_{v_i} + SC_{T_i^{ctr}}$. If $C^u_{v_i} < SC_{v_i} + SC_{T_i^{ctr}}$, the capacitance will be reduced with the insertion of g_i. A power aware clock tree topology with proper buffer and gate insertion will efficiently reduce the switched capacitance, hence cut down the power usage of the circuits.

Given a testcase with the physical location and the activity pattern of all the modules, together with the circuit models of wire/buffer/gate, the objective of our work is to construct a buffered and gated clock network T as well as a controller network T^{ctr}. Subject to the two constraints of zero skew and slew rate constraint, the resultant total switched capacitance $SC_T + SC_{T^{ctr}}$ should be minimized.

6.3 Methodology

In our clock gating work, two synthesizers, HKPUcg and HKPUst, are proposed. HKPUcg applies the same slew rate ($C_d \leq 20 \times C_g$) as previous works [65, 11]. The driver capacitance of every buffer/gate cannot exceed 20 times of a gate input capacitance. Besides, in HKPUst we perform a more strict slew constraint ($\leq 100ps$) along the whole clock network. Slew table is constructed in advance to provide driving ability at each gate/buffer insertion point. We build our clock trees based on two topology generation methods. In HKPUcg, NNS [24] is applied with an newly improved power aware cost function. Every time the pair of nodes with the minimum cost are merged together. In HKPUst, the dual-MST construction method [55] is applied also with newly

improved cost function. The resulting clock tree is close to a full symmetry. As a result of the cost improvement, the according topology can result in lower power usage. A recursive buffer/clock gate insertion method is developed for bottom-up merging. Blockage handling technique from [55] is also involved, because the buffers and gates cannot be placed inside blockage regions. Elmore model [26] is applied for clock delay computation, and DME technique [3, 10] is applied for wirelength minimization. Thus, segment is used instead of point to represent the set of merging location, and deferred embedding is applied to reduce total wirelength.

6.3.1 Power Aware Topology Generation

In order to save the power, the nodes with a bigger similarity of activity patterns should have a higher priority to be matched. Assume v_a and v_b to be a pair of two nodes, as shown in figure 6.2. If the corresponding activity patterns A_a and A_b are similar, the resulted activity A_i will have a shorter active period, and smaller power cost will be caused. Besides the concerns on activity patterns, an estimation of the merging cost $Pwr\,(v_a, v_b)$ is also required. This can be determined in multiple ways. For instance, we can actually merge the two nodes together to obtain the exact connection information. However, exact buffer insertion and wire balancing are performed, which will cost longer time. Instead, we develop a new method for potential switched capacitance estimation. The Manhattan distance between the nodes v_a and v_b is denoted by $D(v_a, v_b)$. The Elmore delay difference of these two nodes is denoted by $DLY(v_a, v_b)$. The delay and power consumption for unit wirelength are denoted by ρ_D and ρ_P respectively, which are computed in advance for simulation reference. If $\frac{DLY(v_a,v_b)}{\rho_D}$ is smaller than $D(v_a, v_b)$, then the two nodes can be merged without snaking wire involved, and the corresponding equation for

power cost computation is shown as below

$$Pwr\left(v_a, v_b\right) = \rho_P \times D(v_a, v_b) \times P\left(A_i\right) \tag{6.2}$$

Otherwise, snaking will be included, as shown in the following equation

$$Pwr\left(v_a, v_b\right) = \rho_P \times \frac{DLY(v_a, v_b)}{\rho_D} \times P\left(A_i\right) \tag{6.3}$$

An improved power aware dual-MST matching technique is developed for HKPUst. A specific definition of a geometric matching of one iteration can be found in [41]. The detailed description is shown in procedure 1 of chapter 5. It is a weighted perfect matching approach. Given a set of nodes $V = \{v_1, v_2 \ldots v_m\}$, we first construct a complete graph $G = \{V, E\}$. Let $|V|$ and $|E|$ denote the number of nodes and edges in the graph G, so$|V| = m$. Since G is a complete graph, every pair of two nodes v_i, v_j is connected by an edge $e_{i,j}$, $E = \{e_{1,2}, e_{1,3} \ldots e_{m-1,m}\}$ and $|E| = \frac{m(m-1)}{2}$. The cost of matching two nodes v_i and v_j is denoted as $f_c\left(e_{i,j}\right)$. Let M denote the matching result of G. M is composed of a group of edges and it is a subset of E. The maximal pairing cost of M is denoted as C_{max} and defined as below. We will get close to a symmetric clock tree by reducing C_{max} in each level. The merging cost $f_c^{DMST}(v_a, v_b)$ for dual-MST is shown as below. α and β are the weights of the Manhattan distance and the estimated power cost, respectively.

$$f_c^{DMST}(v_a, v_b) = \alpha \times D(v_a, v_b) + \beta \times Pwr(v_a, v_b) \tag{6.4}$$

Similarly, an improved power aware NNS matching technique is developed for HKPUcg. We compute the two edges of e_a and e_b based on zero skew concerns. The cost function is an estimation of the resultant total switched capacitance, which is shown as below

$$f_c^{NNS}(v_a, v_b) = |e_a| \times P\left(A_a\right) + |e_b| \times P\left(A_b\right) \tag{6.5}$$

By means of these weighted cost functions, the node pairs with a bigger similarity of switching activity and a shorter distance will have a higher priority to be matched. Our approach of topology generation is based on concurrent gate insertion, therefore the downstream information of the two merging nodes are accurate.

6.3.2 Concurrent Gate and Buffer Insertion

A recursive buffer and gate insertion technique is developed for two objectives: (1) slew rate constraint (2) power usage reduction. Buffers are utilized for power supply to restrict the signal transition time, and clock gate insertion can reduce the switched capacitance by disabling idle sections. We model the buffer and gate with according attributes for Elmore delay computation. Some previous works [15] already proposed to construct a buffered clock tree with zero clock skew. In our work, we apply similar approach for both buffer and clock gate insertion. The input/output capacitance and resistance of the buffers and clock gates should be obtained first. Hence, the delay of wire, buffers and clock gates can be computed based on Elmore RC model.

In HKPUst, we try to maintain the level of buffers and gates of every source-to-sink clock path exactly the same. During the procedure of the bottom-up binary merging, we first examine the two downstream levels of gates. If they differ by two or more, a penalty cost will be engaged. Such matching result will probably be discarded due to the huge cost. Buffer levels will be balanced accordingly. By means of this level balancing, the clock skew will be reduced significantly, and the negative effect caused by signal variation will be reduced.

Here we will describe our technique of gate insertion based on a determined matching result. We first define three different kinds of gate insertion. They are virtual gate insertion at the upstream level, temporal gate insertion at the current level and none gate insertion. Temporal insertion is controlled by the

balancing of gate levels, which will be further divided into two kinds of single gate insertion and one kind of back-to-back double gates insertion. The insertion of a gate is assumed to be closest to the internal merging node for switched capacitance minimization. Since DME technique is applied in our work, we assume the middle point of the merging segment to be the gate location. The comparison among the three assumption of gate insertion are based on the resulting switched capacitance, which are SC_{vir}, SC_{tmp} and SC_{non}, respectively. If the power consumption of the virtual insertion or the none insertion is the smallest, no insertion of any gate will definitely result in less switched capacitance compared to the choice of temporal gate insertion. Therefore, we discard any insertion of gates at the current level. Otherwise, temporal gate insertion will probably reduce the switched capacitance rather than the others, and here we will accept the insertion of gates.

An example is shown in figure 6.2. The activity A_i equals to $A_a \cup A_b$. The edge connection between each of the two nodes to the merging node are denoted as e_a and e_b. C_{e_a} and C_{e_b} are their corresponding capacitance cost. The equations to compute the three resulting switched capacitance are shown as below

$$SC_{vir}(v_a, v_b) = (C_a^u + C_{e_a} + C_b^u + C_{e_b}) \times P(A_i) + C_{T_i^{ctr}}^u \times P_{tr}(A_i) \quad (6.6)$$

$$SC_{tmp}(v_a, v_b) = (C_a^u + C_{e_a}) \times P(A_a) + C_{T_a^{ctr}}^u \times P_{tr}(A_a) + C_b^u + C_{e_b} \quad (6.7)$$

$$SC_{non}(v_a, v_b) = C_a^u + C_{e_a} + C_b^u + C_{e_b} \quad (6.8)$$

Notice that here we only describe the equation of SC_{tmp} for a single gate insertion at node v_a. The other two equations can be derived in a similar way.

6.4 Experimental Results

In this section, our experimental results are presented. We implement our clock network synthesizer in C programming language. The binary is executed on a Linux machine with an Intel Core2 Quad 2.4G Hz CPU and 4GB memory.

We first discuss our work HKPUst. The benchmark suite used for the experiments is released from the ISPD 2009 CNS contest [88]. The detailed information of the benchmark circuits is shown in table 5.2 of chapter 5. In our experiment, one type of wire and one type of buffer is used in our clock tree synthesizer. The unit resistance of the wire is $0.0003\Omega/nm$, and the unit capacitance of the wire is $0.00016fF/nm$. The specific configuration of the buffer in different sizes is shown in table 6.1. In our synthesizer, the maximum buffer size is set to be 6. Hence we list the attributes of different buffer sizes up to 6. Notice that the corresponding attributes of a gate is listed in the last row of table 6.1. This table is generated from our SPICE simulation statistics. C_b means the input capacitance, R_b means the driver resistance and d_b means the internal delay of a buffer, respectively. We also apply SPICE simulation. The according power supply is set to be $Vdd = 1.0V$, and the PTM model is of 45 nanometer scale.

Table 6.1 Buffer and gate configuration

buffer sizes	$C_b\,(fF)$	$R_b(\Omega)$	$d_b(ps)$
1	35	66.9	4.92
2	70	40.5	5.63
3	105	31.3	6.13
4	140	26.4	6.52
5	175	25.0	6.95
6	210	20.7	7.20
gate	35	52.45	17.03

A summary of the performance of HKPUst is shown in table 6.2 and table 6.3. We run our program with different values of α and β for topology

tuning. The clock skew (SKEW), total capacitance (TC), optimal capacitance (OSC), switched capacitance (SC) and CPU time are listed. The respective units are picoseconds (ps) for SKEW, seconds for CPU and femto-farad (fF) for capacitance. TC denotes the original total capacitance cost of the clock tree without gate insertion. OSC denotes the resulted capacitance after disabling of all the idle periods at each node. SC denotes the resulted switched capacitance of our gated clock tree. It can be seen that SC is mostly smaller than TC, which means an effective power reduction in our gated clock tree construction. The nominal skew of each clock tree is zero. Additionally, we use SPICE for further evaluation and get the accurate skew estimation, as listed in the table. The activity pattern of all the sinks are generated by ourselves, according to the instruction and RTL description used in [65]. The length of the activity pattern is 10000 for every benchmark. It can be speculated that the power cost of HKPUst should be larger than the previous ones with loose slew constraint, but the signal transition time is more consistent hence the work is more practical in use. Generally, in HKPUst the switched capacitance can be reduced by around 10% with the insertion of clock gates. Meanwhile, the simulated clock skew is only about 20 ps in average. The runtime of our program is less than 3 seconds, which represents good efficiency.

Besides, we choose another benchmark suite (r1 to r5) from [92] for performance evaluation. In this benchmark suite, the unit resistance of the wire is $0.003\Omega/um$, and the according unit capacitance is $0.02fF/um$. The original benchmarks only contain physical information of the sinks. Further on, they were modified in [65] with the inclusion of logic information (activity patterns). The activity pattern is transformed from the instruction stream based on the logic description of all the instructions. The information of the benchmarks is shown in table 6.4, which is similar to the table 5.11. Additionally, for each benchmark, the number of instructions, the length of the instruction stream

Table 6.2 Clock skew and switched capacitance with gate insertion

Circuits	HKPUst ($\alpha = 1, \beta = 0$)				
	SKEW	TC	OSC	SC	CPU
ispd09f11	20.0	103973	61868	78939	0.37
ispd09f12	17.2	104874	65539	78970	0.34
ispd09f21	20.0	118028	68813	89140	0.35
ispd09f22	15.6	69810	43786	53173	0.32
ispd09f31	33.7	221639	136596	179336	3.83
ispd09f32	33.4	175122	101850	138156	0.51
ispd09f33	20.6	171747	107773	139467	5.44
ispd09f34	22.2	144688	92341	118570	0.49
ispd09f35	16.9	165546	104232	134708	8.11
ispd09fnb1	18.6	32635	23452	32635	0.70
ispd09fnb2	19.7	67041	46550	66280	2.40
avg.	21.6	125009	77527	100852	2.08

and the average activity are listed. For the whole benchmark suite, the average amount of modules, instructions and activity are about 1346.2, 108.2 and 38.9%, respectively. In our experiments, we still use the circuit models of buffers and gates in [65]. Another more practical model is proposed in [11], which is a future work of ours.

We construct another synthesizer, HKPUcg, for performance comparison with previous works. In HKPUcg, the constraint on slew rate is the same as the works in [65] and [11]. That is, the driver capacitance for each buffer/gate cannot exceed $20 \times C_g$. We listed four previous synthesizers with ours, HKPUcg, for performance comparison. GCR and GCRred were proposed in [65]. GCR denotes the gated clock routing algorithm with gate insertion at every internal node of the clock tree. GCRred is an advanced version of GCR, where clock gates are selectively removed for switched capacitance improvement. GBCR and GBCRnt were proposed in [11]. GBCR denotes the gated and buffered clock routing with the application of dynamic programming and solution sampling. Meanwhile, the topology generated in GCR is reused in GBCR. In

Table 6.3 Clock skew and switched capacitance with gate insertion

Circuits	HKPUst ($\alpha = 2, \beta = 1$)				
	SKEW	TC	OSC	SC	CPU
ispd09f11	16.7	103851	61422	78261	0.37
ispd09f12	16.6	103998	65090	79603	0.35
ispd09f21	25.7	108116	67586	81043	0.35
ispd09f22	8.5	69552	43938	53597	0.32
ispd09f31	19.3	220522	128744	174024	5.60
ispd09f32	21.7	162525	103658	123151	0.50
ispd09f33	18.8	155995	100329	128386	6.30
ispd09f34	20.3	139518	88924	109183	0.46
ispd09f35	21.6	163376	102231	128963	8.13
ispd09fnb1	29.6	34370	24869	34370	0.63
ispd09fnb2	27.5	70478	50113	69788	1.90
avg.	20.6	121118	76082	96397	2.26

Table 6.4 Circuit information of the benchmarks from r1 to r5

Circuits	Chip Size (mm x mm)	No. Sinks	No. Inst.	Inst. Length	Average Activity (%)
r1	8.6 x 8.6	267	64	10000	39.86
r2	11.6 x 11.7	598	89	10000	39.60
r3	12.2 x 12.0	862	108	10000	36.91
r4	15.9 x 15.9	1903	120	10000	40.56
r5	18.2 x 17.8	3101	160	10000	37.42
avg.	13.3 x 13.2	1346.2	108.2	10000	38.87

GBCRnt, a new topology was generated for performance improvement. The applied benchmark suite are exactly the same for the above five research works. The performance of the five clock gating research works are shown in table 6.5, table 6.6 and table 6.7. In these two tables, the total switched capacitance cost (SC in pF), total wirelength of the clock tree (CLK in mm) and the controller tree (CTR in mm), and the CPU runtime (CPU in sec) are listed for comparison.

It can be observed that GCR has the largest switched capacitance thus the

Table 6.5 Performance comparison between HKPUcg and other clock gating works.

Circuits	HKPUcg				GCR [65]			
	SC	CLK	CTR	CPU	SC	CLK	CTR	CPU
r1	39.33	1290	1079	0.36	101.12	1320	14784	0.08
r2	84.28	2582	2306	0.57	276.03	2574	46530	0.61
r3	112.53	3305	3086	0.85	405.56	3322	70073	1.50
r4	241.55	6795	5588	2.29	1104.28	6671	203585	5.81
r5	372.19	9856	9802	6.14	2037.13	10072	393876	18.8
avg.	169.98	4766	4372	2.04	784.28	4792	145770	5.36

Table 6.6 Performance comparison between HKPUcg and other clock gating works.

Circuits	GCRred [65]				GBCR [11]		
	SC	CLK	CTR	CPU	SC	CLK	CTR
r1	50.95	1581	3597	0.09	46.81	1657	1400
r2	115.35	3101	10937	0.61	98.63	3328	2985
r3	157.47	4130	15767	1.52	141.66	5322	4831
r4	367.08	8918	43306	5.90	369.28	13568	8855
r5	590.84	13058	78879	19.03	572.82	20543	12588
avg.	256.34	6158	30497	5.43	245.84	8884	6132

power consumption. The major reason is every node in GCR is assigned with a clock gate, therefore the resultant controller tree becomes the main contribution to the total power usage. With the inclusion of gate removal technique, GCRred effectively reduces 79.1% of the total wirelength of its controller tree. Meanwhile, the addition in clock tree is acceptable (only 28.5%). As a matter of fact, the resulting cost of the switched capacitance in GCRred is reduced by 67.3% compared to GCR. Despite the performance improvement, the heuristic of gate removing in GCRred is generally based on the activity comparison, which is not fully performance-driven. There are still useless gate insertion which may cause negative effect towards the power reduction.

Table 6.7 Performance comparison between HKPUcg and other clock gating works.

Circuits	GBCRnt [11]		
	SC	CLK	CTR
r1	40.36	1357	675
r2	85.86	2713	1863
r3	121.17	3828	1973
r4	249.82	7395	3358
r5	377.87	10520	4211
avg.	175.02	5163	2416

GBCR is implemented based on the objective of switched capacitance minimization as well as the modified benchmark suite proposed in [65]. Buffer insertion is applied to replace some of the gate insertion in GCRred for driving power supply. As a result, the total wirelength of the according controller tree is further reduced by 79.9% compared to that of GCRred. When two subtrees are merged together, a group of $25K^2$ merging segments are generated according to different gating/buffering scenarios. In experiments of GBCR, K is kept to be 10 for every simulation. Thus there are 2500 times of delay and segment computation in every merging procedure, which is much more time consuming. In GBCR, the topology from GCRred is directly reused, and the resulting switched capacitance is reduced by 4.1%. The topology generation is based on NNS [24], with the cost function to be the resultant switched capacitance for the merging. The formula is shown as below

$$f_{a,b}^{GCR} = (C_a^u + C_a^n) \times p_a + (C_b^u + C_b^n) \times p_b + C_a^{ctr} \times p_a^{tr} + C_b^{ctr} \times p_b^{tr} \qquad (6.9)$$

GBCRnt is an improved version of GBCR, where a new topology is devised. It is also based on NNS [24] with newly developed cost function involved. The formula is shown as below

$$f_{a,b}^{GBCRnt} = |e_a| \times p_a + |e_b| \times p_b + \beta \times \left(|e_a^{ctr}| \times p_a^{tr} + |e_b^{ctr}| \times p_b^{tr}\right) \quad (6.10)$$

where β is set to be 0.01 in the experiments. The cost function is only used for nodes matching, and the controller tree may not have the edges e_a^{ctr} and e_b^{ctr} indeed. Therefore, β should be less than 1.0. With the inclusion of the new topology, GBCRnt has improvement on both clock tree and the clock tree. Compared to GBCR, the wirelength of the clock tree and the controller tree have been reduced by 41.9% and 60.6%, accordingly. As a matter of fact, the resultant total switched capacitance is reduced by 28.8%.

Compared to the best reported experimental results, that is, the results from GBCRnt, our work has bigger wirelength of clock and controller tree. However, the resulting total switched capacitance is smaller. This is because the the efficiency of each gate for masking off idle branches is higher in our synthesizer, and the nodes with similar activity patterns are more probable to be merged. Our synthesizer, HKPUcg, proposes better performance compared to the other four gating works. Generally, HKPUcg can reduce the switched capacitance by 78.3%, 33.7%, 30.9% and 2.9%, compared to GCR, GCRred, GBCR and GBCRnt, respectively. GBCRnt is the best reported clock gating work so far, and our synthesizer can still outperform it. Meanwhile, there are 2500 times of computation for every merging procedure. However, in HKPUcg there are only 5 times of computation accordingly. Thus there is a great improvement on the efficiency in HKPUcg.

Let K denote the amount of instructions, L denote the maximum number of active sinks of all the instructions, B denote the length of the instruction stream, and N denote the number of sinks. As stated in [65],the time complexity is composed of $O(B)$ for initialization, $O(NKL)$ for activity computation, $O(N^2K^2)$ for transitional probability computation and (N^2) for topology generation. As a matter of fact, the overall complexity is $O(B + N^2K^2)$. Notice

that N^2 is only the average complexity of topology generation, the complexity of the worst case is still N^3. The complexity of the work in [11] is stated as $O(N^3 + BN)$, which is worse with the operation on the whole activity pattern at each merging node.

In our work, we use a $K \times K$ matrix to restore the instruction transition-module activation table (ITMAT [65]) for every sink, the total additional memory cost is NK^2. During the bottom-up merging procedure, it is just necessary to compute the OR result of the ITMAT of two nodes, with the complexity of K^2. Therefore, the total complexity for transitional probability computation is only NK^2. Notice that the ITMAT matrix of any nodes can be reused by their parent nodes, so the total extra cost on memory is still NK^2. As a result, the overall complexity of our clock gating work is reduced to $O(B + N^2 + NK^2)$.

The experimental results regarding the major three metrics (SC, CLK and CTR) of GCR, GCRred, GBCR and GBCRnt are all obtained from [11] (the results of GCR and GCRred reported in [65] have some errors in the computation of switched capacitance, which are verified and corrected in [11]). Additionally, we obtain the debugged source code of the work in [65]. For the comparison of runtime, we execute the according program of GCR and GCRred in our platform. In table 6.5 and table 6.6, the CPU runtime of HKPUcg, GCR and GCRred is also listed. It is shown that due to the improvement on complexity of our program (from $O(B + N^2K^2)$ to $(B + NK^2 + N^2)$), the average runtime is reduced by 61.9%, compared to the runtime of GCR and GCRred. The runtime of GBCR and GBCRnt have not been obtained for comparison.

6.5 Summary

Power saving and clock skew are two major concerns in clock network synthesis. A power aware topology generation is proposed with improved design on the cost function. Meanwhile, a concurrent buffer/gate insertion technique is

proposed. It can distribute the gate properly in high efficiency. This is developed in order to optimize the power dissipation of a clock distribution network. In both of the two proposed synthesizers, HKPUcg and HKPUst, zero clock skew is guaranteed. Experimental results show that our method can greatly reduce the switched capacitance hence reduce the power consumption of the clock network. Compared to the previous clock gating works, HKPUcg have better performance on power saving. Meanwhile, the resultant simulated clock skew of HKPUst under SPICE is still acceptable, with no slew rate violation engaged.

Chapter 7

Global Routing

7.1 Overview

The purpose of a global router is to decompose a large routing problem into small and manageable subproblems (detailed routing) [79]. Our router, HKPUgr, can be divided into two main parts: 2D global routing and 3D layer assigning. In the 2D global routing, it is composed of two steps: original routing and iterative rerouting. During the first step, we apply FLUTE [19, 20] to get the steiner points of each net, Minimum spanning tree (MST) is then constructed to generate the original network. Based on the according topology, each net is decomposed into 2-pin nets, and routed with L-shape routing technique. The net ordering of L-shape routing is in an increasing order of the bounding box size. In the following step of iterative rip-up and rerouting, we do L-shape rerouting for M_L times followed by monotonic rerouting for M_M times. Maze rerouting is then engaged to further improve the congestion with three substeps involved. They are congested-cost-function based maze rerouting, historical-cost-based maze rerouting and surrounding nets rerouting. When the target on total overflow is reached, we use maze routing once more to reduce the wirelength. Notice that no additional congestion will be further caused.

Table 7.1 Notations in global routing

Notation	Description
n_i	The ith net of the net list
p_i	The ith pin in a net
s_i	The ith steiner point in a net
K_{p_i},K_{s_i}	The node structure of the ith pin or steiner point
$sub(p_i, p_j)$	The subnet between p_i and p_j
G_i	The ith grid point
$E_{(i,j)}$	The edge between G_i and G_j
c_{G_i}	The accumulated cost of G_i
$c_{E_{i,j}}$	The accumulated cost of $E_{i,j}$
$c_{(G_i,G_j)}$	The cost of the edge between G_i and G_j
c_{bnd}	The unit cost of a bending
$F_E(i)$	The cost of the edge E with usage of i
$C_E(i)$	The first order of $F_E(i)$
p_i	The probability to be occupied with usage of i

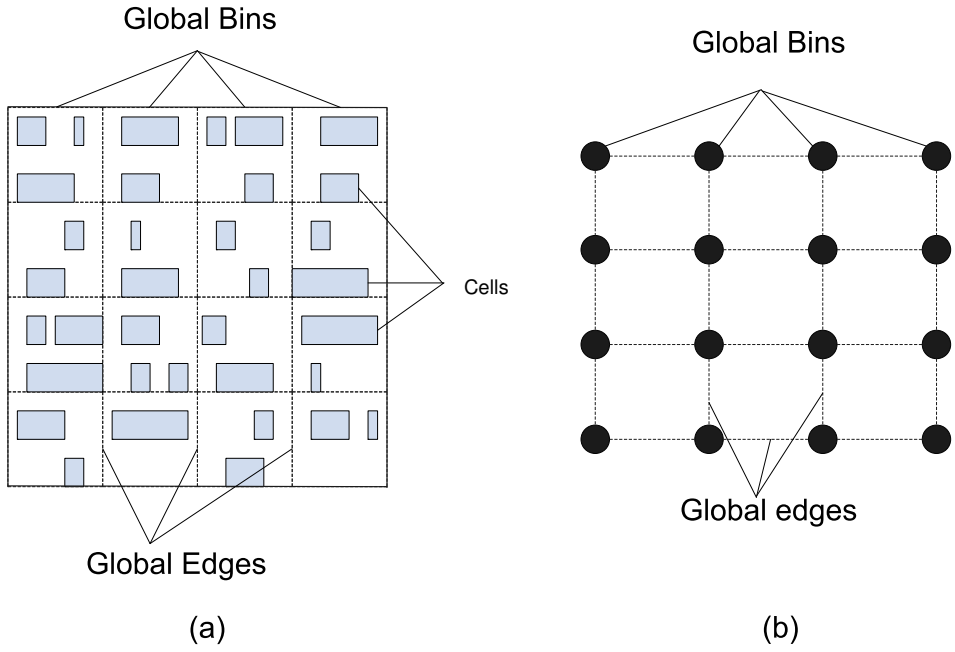
7.2 Problem Formulation

The chip is decomposed into rectangular global bins of the same size. All the pins are then distributed into their according bins. An example of chip decomposition is shown in figure 7.1(a). In figure 7.1(b), each vertex represents a global bin (grid point), and each edge (grid edge) connecting two adjacent bins represents the boundary in figure 7.1(a). In global routing, only inter-bin connections are implemented. The implementation of inner-bin connections are completed in detailed routing.

Let N denote the set of nets, $N = \{n_1, n_2, \ldots, n_{|N|}\}$. Let P_i denote the set of pins for the ith net, $P_i = \{p_i^1, p_i^2, \ldots, p_i^{|P_i|}\}$. We use e_i to denote the edge i, and the set of edges is denoted by E. We use OF_{e_i} to denote the overflow on the edge e_i. c_{e_i} and d_{e_i} are used to denote the capacity and demand of edge e_i, respectively. On the edge e_i, the overflow is defined as the exceeding amount of demand towards its capacity. If d_{e_i} is greater than c_{e_i}, OF_{e_i} equals $d_{e_i} - c_{e_i}$. Otherwise, OF_{e_i} equals zero.

The global routing problem is usually formulated as below. Given a grid

Figure 7.1 (a) Chip decomposition (b) Grid graph



graph as shown in figure 7.1(b), a set of networks N and the capacity of edges E, the major task of a global router is to implement the inter-bin connection for each network. The resultant overflow is the major metric and it should be minimized. The total wirelength, as a secondary concern, should also be minimized.

7.3 Methodology

7.3.1 Dynamic Steiner Point Relocation

In global routing, rip-up and rerouting is a popular approach. In each iteration, nets or 2-pin subnets are selectively rip-up and rerouted for one time. As this procedure proceeds, congestion and overflow will be improved at later iterations. Steiner points are originally determined by FLUTE [19, 20] in most

global routers. However, during the procedure of iterative rip-up and rerouting, topology of the network improved and the steiner points need relocating. Some routers rebuild the topology at the end of each iteration for steiner points regeneration. However, this method is restrictive to the space of improvement. If there are several subnets of a network to be rerouted in one iteration, unchanged steiner point after the rerouting of one subnet may damage the quality of the rerouting of other subnets. Meanwhile, since tree traversal is needed for steiner points regeneration, it is also time consuming.

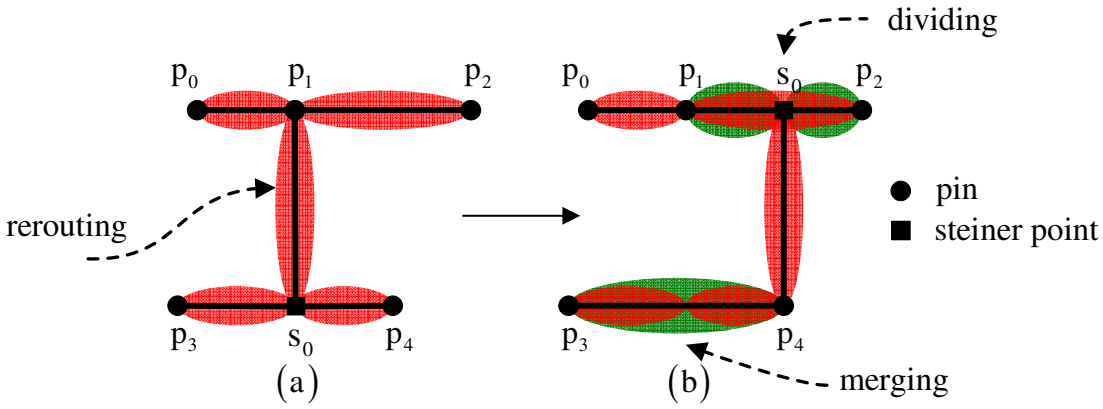
In our router, we develop a new technique, dynamic steiner point relocation, to tackle the problem. After rerouting the subnet sub_j, we dynamically determine and relocate the steiner points of the net to the optimal location. Here we use nodes to represent both pins and steiner points. For each node, it has at least one but at most four neighboring nodes. We use a data structure K to record the information of nodes, including coordinates, address of neighbouring nodes and corresponding subnets. Notice that the neighborhood is a mutual relationship.It is quite easy to traverse the whole net from any starting node in the net.

As shown in figure 7.2(a), a net n_i is assumed to be composed of five pins and one steiner point, with $K_{p_0}, \ldots, K_{p_4}, K_{s_0}$ as their corresponding node structures. We can see that K_{p_1} is the only neighbor of K_{p_0}, with the connecting subnet $sub(p_0, p_1)$. K_{p_1} has three neighbors of $K_{p_0}, K_{p_2}, K_{s_0}$. In this example, $sub\,(p_1, s_0)$ is the subnet to be rip-upped and rerouted. Before rerouting, we traverse other subnets and label the global bins with according subnet id. This method can facilitate the generation of new steiner point after rerouting. After the rerouting of $sub\,(p_1, s_0)$ and the formation of the new connection, the two ending nodes become s_0' and p_4, as shown in 7.2(b). s_0 is thus removed, and the two subnets $sub\,(p_3, s_0)$ and $sub\,(p_4, s_0)$ are merged together to become one subnet $sub\,(p_3, p_4)$. Meanwhile, K_{p_4} has replaced K_{s_0} to be the neighbor of K_{p_3}, vice versa. A new steiner point s_0' is then generated on the path between

p_1 and p_2. Subnet $sub\,(p_1, p_2)$ is divided into $sub\,(p_1, s_0')$ and $sub\,(p_2, s_0')$, and $K_{s_0'}$ becomes the common neighbor of both K_{p_1} and K_{p_2}.

Assume it to be the ith iteration of the rerouting. In this example, $sub\,(p_1, s_0)$ is transformed into $sub\,(p_4, s_0')$, and it is marked with the iteration label i. Besides, $sub\,(p_3, p_4)$, $sub\,(p_1, s_0')$ and $sub\,(p_2, s_0')$ are newly generated and we need to determine whether or not to reroute them in the same iteration. There are additional judgment on condition that the requirement is met. If both (p_3, s_0) and $sub\,(p_4, s_0)$ have been rerouted in the ith iteration, $sub\,(p_3, p_4)$ will be rerouted with a smaller probability K_0, because it is a result of merging and the potential of improvement is expanded. Otherwise, $sub\,(p_3, p_4)$ will be rerouted with a bigger probability K_1. On the contrary, owing to the reduced freedom of solution, $sub\,(p_1, s_0')$ and $sub\,(p_2, s_0')$ should be rerouted only when $sub\,(p_1, p_2)$ has not been rerouted in this iteration. Additionally, a probability which is quite close to one is multiplied to the decision value in case deadlocks.

Figure 7.2 (a) Before rerouting (b) After rerouting



7.3.2 Edge-based Monotonic and Maze Routing

In global routing, layer assignment is applied after completing the planar inter-bin connection of all the networks. If the edges of the planar connection of a network are assigned into different layers, inter-layer connection (via) is necessary to keep the connectivity of the network.

In the planar view, the resultant amount of via for each network is generally determined by its number of planar bendings. In the modern global routers, the factor of bendings is usually involved in the cost function to improve the performan of routing and rip-up rerouting.

Traditional maze routers are based on the propagation of grid points, and it utilize a queue to maintain the points which have been reached. In each step, the point with the minimal cost is popped from the queue, and we call it the major point. The the adjacent four points are propagated from the major point, and each of them is labelled with an accumulated cost value, as well as a parent point. Such procedure continues until we reach the sink terminal. This approach can get the optimal result, if the cost is a sum of the two weighted factors, congestion and wirelength. Nonetheless, with the amount of bendings as an additioanl factor, the result generated by the maze router is no longer optimum.

In the global routing, the cost function is usually composed of the following three factors: congestion, wirelength and bendings, shown as below

$$Cost = \#cong + \#wire + \#bend \tag{7.1}$$

In this section, we present an example to describe the shortcomings of maze router based on point propagation. As shown in figure 7.3 (a), $\{G_0, \ldots G_3\}$ denotes the grid points, and c_{G_i} denotes the cost of G_i. Suppose that G_0 and G_1 have already been reached, with cost values c_{G_0} and c_{G_1} accordingly. At the point G_2, there are two solutions available. $c^0_{G_2} = c_{G_0} + c_{(G_0,G_2)}$ and $c^1_{G_2} =$

$c_{G_1}+c_{(G_1,G_2)}$. Assume that $c^0_{G_2} > c^1_{G_2}$, then G_2 is labelled with value of $c^1_{G_2}$, and G_1 becomes the parent of G_2. In the next turn, G_3 is propagated from G_2 with a cost of $c_{G_3} = c^1_{G_2} + c_{bnd} + c_{(G_2,G_3)}$. This procedure appears very reasonable, because we get the solution $(G_3 \rightarrow G_2 \rightarrow G_1)$ for G_3. However, due to the additional bending cost, it is possible that $c^0_{G_2} + c_{(G_2,G_3)} < c^1_{G_2} + c_{bnd} + c_{(G_2,G_3)}$. Therefore, the path $(G_3 \rightarrow G_2 \rightarrow G_0)$ is a better choice than $(G_3 \rightarrow G_2 \rightarrow G_1)$, but it cannot be acquired through traditional maze routing approach.

We develop a new maze router based on the propagation of grid edges, optimum result based on equation 7.1 can always be obtained. $E_{(i,j)}$ denotes the edge connecting the points G_i and G_j, and $c_{E_{i,j}}$ denotes the respective propagation cost. In our method, we use edge as the basic propagation unit. During the routing, each edge in the grid is assigned with a cost value and a parent edge. The edge cost is also an accumulated value from the source edge. We apply a priority queue as a storage for the edges. During each time of rerouting, the target two pin net will be removed, and the net is decomposed into to two subnets : a source subnet and a sink subnet. The basic object of rerouting is to connect the two subnets together. At the beginning, we traverse the source subnet and store all the the adjacent edges into the queue. In each step of propagation, one edge is extracted from the top of the queue, which has the minimal cost value. This edge has one of its two adjacent points as the starting point, and the other one as the ending point. The other three edges adjacent to the ending point are the target edges for propagation. Notice that the one of the three with the same direction as the original edge is assured to have the optimal solution. However, the other two perpendicular edges would possibly be updated with better solution in the following steps, due to the bending factors.

An example of edge propagation is shown in figure 7.4(a). Suppose that $E_{(0,2)}$ and $E_{1,2}$ have already been reached and stored in the queue. Suppose that $E_{0,2}$ is currently the major edge, and $E_{1,2}$ is still in the queue, thus

$c_{E_{0,2}} < c_{E_{1,2}}$. Two edges $\{E_{(2,4)}, E_{(2,3)}\}$ can be propagated in this turn. $E_{(2,3)}$ is in the same direction with $E_{(0,2)}$, so $c_{E_{2,3}} = c_{E_{0,2}} + c_{G_2,G_3}$ is determined to be the accumulated cost value of $E_{2,3}$. Notice that no further update for $E_{(2,3)}$ is necessary, because any cost c_x from the following major edges will be larger than $c_{E_{0,2}}$, therefore $c_x + c_{G_2,G_3} > c_{E_{0,2}} + c_{G_2,G_3}$. Therefore, $E_{0,2} \rightarrow E_{2,3}$ is proved to be the optimal solution for $E_{2,3}$. Meanwhile, $E_{2,4}$ is in the perpendicular direction with $E_{0,2}$, and $c'_{E_{2,4}} = c_{E_{0,2}} + c_{G_2,G_4} + c_{bnd}$ become the cost of $E_{2,4}$, and $E_{0,2} \rightarrow E_{2,4}$ is a conditional solution for $E_{2,4}$. In the next step, $E_{1,2}$ becomes the major edge, as shown in figure 7.4(b). If $c'_{E_{2,4}} > c_{E_{1,2}} + c_{E_{2,4}}$, $E_{1,2} \rightarrow E_{2,4}$ turns out to be a better solution than $E_{0,2} \rightarrow E_{2,4}$, and it is also the optimal solution for $E_{2,4}$. $E_{2,4}$ is then updated with the parent path $E_{1,2}$ and the respective cost, and it won't be updated in future. Generally, in the new approach the optimal solution can be certainly reached, and the computational cost for updating is reduced compared to the traditional maze router.

Figure 7.3 Comparison of two solutions

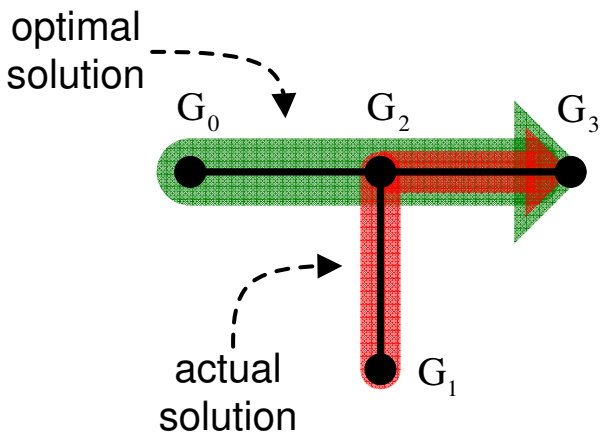
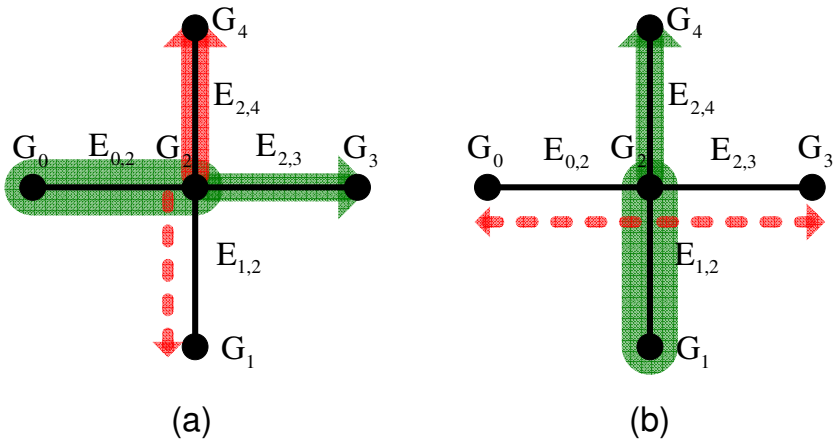


Figure 7.4 (a) Global bin-based router (b) Global edge-based router



7.4 Experimental Results

In this section, our experimental results are presented. We implement our router in C programming language and the router is executed on Linux operating system with a Intel Xeon 1.6GHz CPU and 16GB memory. Besides 2-D router, we use the layer assignment work from NCTU [21] to build up the 3-D solution. The benchmark circuits used in our experiment are proposed in [61]. The detailed information of the benchmark circuits are shown in table 7.2.

Based on the ISPD08 benchmarks, we compare the results of our router HKPUgr with the previous research works. NTHU-Route2.0 [9], NTUgr [33] and FastRoute3.0 [101] are the winners in the ISPD 2008 Global Routing Contest [38]. Besides, NCTU [21] is also included for comparison, which is the best reported work so far. The experimental results of various routers are shown in table 7.3, table 7.4 and table 7.5. It can be found that our router HKPUgr can outperform all the other works in 6 benchmarks, with less overflow and shorter wirelength caused. Meanwhile, our router HKPUgr can solve 12 cases without any overflow generated.

Additionally, we considered the robustness of the global routers. Specified

parameter settings may deteriorate the robustness of the routers, and these will make the routers unadaptable towards other cases. As a matter of fact, we modified the benchmarks with different capacity supplies. The corresponding statistics will become a verification of the stability of different routers. In table 7.6, 4 units of capacity (2 tracks) are removed from each edge, in order to make the benchmarks more congested. Similarly, In table 7.7 4 units of capacity (2 tracks) are added to each edge so as to alleviate the congestion. We include the best reported global router, NCTU, for comparison based on those modified benchmarks. From the result of worse congestion in table 7.6, our router is better in 13 cases. It represents the adaptability of our router in the congested situation. From the result of less congestion in table 7.7, our router can outperform the other in 6 cases.

Table 7.2 Information of ISPD08 benchmarks

Circuits List	Grids Size	No. of nets	No. of routed nets	Max. deg.	Ave. deg.
adaptec1	324x324	219k	177k	340	4.2
adaptec2	424x424	260k	208k	153	3.9
adaptec3	774x779	466k	368k	82	4.0
adaptec4	774x779	515k	401k	171	3.7
adaptec5	465x468	867k	548k	121	4.1
newblue1	399x399	332k	271k	74	3.5
newblue2	557x463	463k	374k	116	3.6
newblue3	973x1256	552k	442k	141	3.2
bigblue1	227x227	283k	197k	74	4.1
bigblue2	438x471	577k	429k	260	3.5
bigblue3	555x557	1.12M	666k	91	3.4
bigblue4	403x405	2.23M	1.13M	129	3.7
newblue4	455x458	636k	531k	152	3.6
newblue5	637x640	1.26M	892k	258	4.1
newblue6	463x464	1.29M	835k	123	3.8
newblue7	488x490	2.64M	1.65M	113	3.6

Table 7.3 Performance comparison based on ISPD08 benchmarks

	HKPUgr			NTHU-Route2.0 [9]		
Circuits	OVF	WL	CPU	OVF	WL	CPU
adaptec1	0	53.33	885	0	53.44	600
adaptec2	0	51.76	181	0	52.29	126
adaptec3	0	129.7	1164	0	131.0	654
adaptec4	0	120.6	206	0	121.7	156
adaptec5	0	153.7	1419	0	155.4	1380
bigblue1	0	56.40	970	0	56.00	842
bigblue2	0	90.09	550	0	90.60	520
bigblue3	0	129.8	398	0	130.7	300
bigblue4	152	224.5	6383	162	231.0	7896
newblue1	0	46.13	655	0	46.50	372
newblue2	0	74.69	93	0	75.70	72
newblue3	34960	105.3	16473	31454	106.5	7344
newblue4	158	127.6	3071	138	130.5	5802
newblue5	0	228.9	1562	0	231.6	1164
newblue6	0	176.0	1402	0	176.9	1086
newblue7	78	348.1	6738	62	353.4	8634

7.5 Summary

In general, we develop a new global router HKPUgr to improve the routing quality. Aside from the traditional techniques, we introduce two new methods for better performance, including dynamic steiner points relocation and edge-based monotonic and maze routing. Our router is implemented in the planar view, and the routing result will be projected into 3-dimension stereo view to assign the connection into different layers accordingly.

Table 7.4 Performance comparison based on ISPD08 benchmarks

	NTUgr [33]			FastRoute3.0 [101]		
Circuits	OVF	WL	CPU	OVF	WL	CPU
adaptec1	0	57.40	270	0	55.20	300
adaptec2	0	53.70	66	0	53.70	60
adaptec3	0	135.0	264	0	133.0	240
adaptec4	0	123.7	72	0	123.0	60
adaptec5	0	159.9	918	0	161.0	660
bigblue1	0	60.00	1086	0	59.50	420
bigblue2	0	91.20	14898	0	98.80	960
bigblue3	0	133.5	240	0	132.0	120
bigblue4	188	242.8	24786	156	244.0	1800
newblue1	6	49.30	58650	0	48.20	300
newblue2	0	76.90	36	0	76.30	60
newblue3	31024	188.3	50640	31634	132.0	2160
newblue4	142	143.8	67086	154	154.0	1020
newblue5	0	244.9	1230	0	240.0	660
newblue6	0	186.6	1278	0	186.0	600
newblue7	310	372.2	86730	108	361.0	9600

Table 7.5 Performance comparison based on ISPD08 benchmarks

	NCTU [21]		
Circuits	OVF	WL	CPU
adaptec1	0	53.5	234
adaptec2	0	51.69	87
adaptec3	0	130.35	293
adaptec4	0	120.67	137
adaptec5	0	154.7	544
bigblue1	0	56.56	381
bigblue2	0	89.4	671
bigblue3	0	129.66	263
bigblue4	164	223.99	3922
newblue1	0	45.99	218
newblue2	0	74.88	54
newblue3	31808	104.28	7886
newblue4	134	126.79	2455
newblue5	0	230.31	902
newblue6	0	176.87	580
newblue7	114	338.63	4291

Table 7.6 Comparison of HKPUgr and NCTU on the modified ISPD08 benchmarks, with 2 tracks removed on each edge.

	-4					
	HKPUgr			NCTU [21]		
Circuits	OVF	WL	CPU	OVF	WL	CPU
adaptec1	44	55.56	2223	4618	56.88	648
adaptec2	706	52.78	1592	3816	52.43	338
adaptec3	0	131.9	2559	4	133.5	867
adaptec4	0	121.2	442	0	121.2	283
adaptec5	0	156.8	2649	2298	158.1	1483
bigblue1	1922	58.74	2559	12046	58.32	916
bigblue2	4782	95.63	3508	4382	93.24	1527
bigblue3	24	130.3	1239	760	130.2	718
bigblue4	730	226.9	9980	870	226.4	8586
newblue1	3366	47.73	2760	4224	47.25	553
newblue2	0	74.75	152	0	74.99	83
newblue3	38562	106.0	17200	NA	NA	NA
newblue4	1902	130.7	7758	1800	129.3	5671
newblue5	482	232.3	6880	1994	232.5	2619
newblue6	0	180.1	3452	8912	181.0	1804
newblue7	6750	353.9	38789	NA	NA	NA

Table 7.7 Comparison of HKPUgr and NCTU on the modified ISPD08 benchmarks, with 2 tracks added on each edge.

	+4					
	HKPUgr			NCTU [21]		
Circuits	OVF	WL	CPU	OVF	WL	CPU
adaptec1	0	52.31	635	0	52.30	253
adaptec2	0	51.21	138	0	51.13	99
adaptec3	0	128.4	953	0	128.8	356
adaptec4	0	120.3	218	0	120.3	193
adaptec5	0	152.0	1223	0	152.4	648
bigblue1	0	54.42	637	0	54.80	341
bigblue2	0	87.70	334	0	87.37	440
bigblue3	0	129.3	439	0	129.2	307
bigblue4	0	224.3	788	0	223.1	1303
newblue1	0	45.14	124	0	45.03	154
newblue2	0	74.64	124	0	74.83	81
newblue3	32048	104.9	16112	28920	104.0	13599
newblue4	0	126.1	469	0	125.2	749
newblue5	0	227.0	1334	0	227.0	1561
newblue6	0	173.6	1119	0	173.8	610
newblue7	0	343.7	931	NA	NA	NA

Chapter 8

Conclusion

In this thesis, we have introduced some fundamental ideas regarding the research topic of routability of the VLSI physical design. Specifically, our focus is located at congestion prediction, clock network synthesis, clock gating design and global routing. In the literature review at chapter 3, we can see that plenty of research works have been proposed by earlier researchers to address the problem of routability. Based on the previous achievements, our work has proposed novel techniques to further improve the performance and enhance the routability.

Congestion prediction is applied at early stages of the physical design. It provides the designer with an estimation of the routability on the chip. Based on the analysis of previous estimation models, we develop three models in chapter 4 for accuracy improvement. SMD model is the fastest technique with the least amount of computation involved, only the tiles inside the bounding box of each subnet are included. Further on, detour model pays extra attention to the surrounding tiles, which becomes adaptive to the over-congested areas. 3-step approach is composed of routing selection and congestion redistribution. The result is of enhanced similarity with that of the technique of iterative rip-up and rerouting, which is widely applied in the modern global routers. From the experimental results, it can be concluded that our models have made significant improvements on the accuracy of prediction, compared

the the previous models.

In terms of the requirements proposed in the ISPD 2009 CNS contest, we develop two clock network synthesizers (DMST and DMSTSS) in chapter 5 with six novel techniques involved. A new method of topology construction, dual-MST, is proposed to build up a tree structure that is very close to a symmetric one. This approach can reduce the sensitivity of the clock skew against voltage variation effectively. Moreover, the developments on hierarchical buffer sizing, iterative buffer insertion and blockage handling techniques further improve the performance and reduce the power consumption. From slew table, the signal transition time can be reduced by the reference of driver ability. In DMSTSS, a method of merging point relocation is developed. Based on the delay estimation of SPICE simulation, clock skew is further reduced by the relocation. Experimental results show that our DMSTSS can outperform all the previous synthesizers with smaller CLR and program runtime in every benchmark. Meanwhile, the three constraints on slew rate, buffer distribution and total capacitance are followed.

For the objective of power saving in the clock network, enough concerns should be taken during the procedure of synthesis. In our work, we achieve the target of power saving by logic gates insertion to mask off the inactive modules and branches. We propose two gated synthesizers (HKPUcg and HKPUst) in chapter 6 with several novel techniques. Improved implementation on topology generation are proposed. Besides, a newly developed buffer/gate insertion technique are devised to reduce the power dissipation. Compared to the previous clock gating works, the resultant switched capacitance of HKPUcg is smaller. Meanwhile, the complexity of our algorithm is smaller, which is shown by the reduced CPU runtime in HKPUcg. In HKPUst, the buffered/gated solution can reduce the switched capacitance compared to that of its original buffered solution, and the resulting clock skew is still acceptable.

In the stage of global routing, routability is fully emphasized. A new global

router, HKPUgr, is proposed in chapter 7 aiming at a better quality of routing design. Our router is built up upon the technique of iterative rip-up and rerouting. Furthermore, we introduce two new methods to improve the performance, including dynamic steiner points relocation and edge-based monotonic and maze routing. With fast reduction on overflow, the amount of wirelength and via connection can also be reduced by our approach. From the experimental results, we can see that most of the benchmarks can be solved by our approach with a shorter wirelength.

Routability has dominant effect on the quality of signal transmission and the timing delay of the circuit. Therefore, exclusive concerns should be involved throughout the whole VLSI physical design procedure. In particular, we achieve the improvement on routability in several correlative steps. During the early designing time, our prediction models will provide the designer with a good estimation on the routability of the packing results, so as to facilitate routing the networks in the subsequent routing procedure. In clock network synthesis, our contribution on proper buffer distribution will maintain the routability of the clock tree. Additionally, clock gating is an associating step of CNS aiming at power reduction. Subsequently, global routing implements the connection of the rest of the networks inside the area of the chip. The newly proposed techniques will shorten the total wirelength, in order to reduce the signal delay and improve the routability.

Bibliography

[1] C. Albrecht. Global Routing by New Approximation Algorithms for Multicommodity Flow. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(5):622 – 631, May 2001.

[2] H. Anway, G. Farnham, and R. Reid. Plint Layout Systems for VLSI Chips. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 449 – 452, 1985.

[3] K. D. Boese and A. B. Kahng. Zero-Skew Clock Routing Trees with Minimum Wirelength. In *Proceedings of 5th the Annual IEEE International ASIC Conference and Exhibit*, pages 17 – 21, 1992.

[4] M. Burstein and R. Pelavin. Hierarchical Wire Routing. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 223 – 234, 1983.

[5] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Can Recursive Bisection Produce Routable Placements? In *Proceedings of IEEE/ACM Design Automation Conference*, pages 477–482, 2000.

[6] R. C. Carden and C. K. Cheng. A Global Router Using an Efficient Approximate Multicommodity Multiterminal Flow Algorithm. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 316 – 321, 1991.

[7] C. C. Chang, J. Cong, D. Z. Pan, and X. Yuan. Interconnect-Driven Floorplanning with Fast Global Wiring Planning and Optimization. In *Proceedings of SRC Tech. Conference*, November 2000.

[8] C.-M. Chang, S.-H. Huang, Y.-K. Ho, J.-Z. Lin, H.-P. Wang, and Y.-S. Lu. Type-Matching Clock Tree for Zero Skew Clock Gating. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 714–719, June 2008.

[9] Y.-J. Chang, Y.-T. Lee, and T.-C. Wang. NTHU-Route 2.0: A Fast and Stable Global Router. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 338 – 343, November 2008.

[10] T. H. Chao, Y. C. Hsu, J. M. Ho, and A. B. Kahng. Zero Skew Clock Routing with Minimum Wirelength. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing.*, 39(11):799 – 814, 1992.

[11] W. C. Chao and W. K. Mak. Low-Power Gated and Buffered Clock Network Construction. *ACM Transactions on Design Automation of Electronic Systems*, 13(1), January 2008.

[12] R. Chaturvedi and J. Hu. Buffered Clock Tree for High Quality IC design. In *International Symposium on Quality Electronic Design*, pages 381 – 386, 2004.

[13] C. Chen, C. Kang, and M. Sarrafzadeh. Activity-Sensitive Clock Tree Construction for Low Power. In *International Symposium on Low Power Electronics and Design*, pages 279 – 282, 2002.

[14] H. M. Chen, H. Zhou, F. Y. Young, D. Wong, H. H. Yang, and N. Shervani. Integrated Floorplanning and Interconnect Planning. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 354 – 357, November 1999.

[15] Y. P. Chen and D. F. Wong. An Algorithm for Zero-Skew Clock Tree Routing with Buffer Insertion. In *Proceedings of European Design and Test Conference*, pages 230–236, 1996.

[16] C. K. Cheng and E. S. Kuh. Module Placement Based on Resistive Network Optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 3:218 – 225, 1984.

[17] J. D. Cho and M. Sarrafzadeh. A Buffer Distribution Algorithm for High-Speed Clock Routing. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 537–543, June 1993.

[18] M. Cho and D. Z. Pan. BoxRouter: A New Global Router Based on Box Expansion and Progressive ILP. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 373 – 378, July 2006.

[19] C. Chu. FLUTE: Fast Lookup Table Based Wirelength Estimation Technique. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 696 – 701, November 2004.

[20] C. Chu and Y.-C. Wong. FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(1):257 – 268, January 2008.

[21] K.-R. Dai, W.-H. Liu, and Y.-L. Li. Efficient Simulated Evolution Based Rerouting and Congestion-Relaxed Layer Assignment on 3-D Global

Routing. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 570–575, January 2009.

[22] M. Donno, A. Ivaldi, L. Benini, and E. Macii. Clock-Tree Power Optimization based on RTL Clock-Gating. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 622 – 627, June 2003.

[23] A. E. Dunlop and B. W. Kernighan. A Procedure for Placement of Standard Cell VLSI Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 4:92 – 98, 1985.

[24] M. Edahiro. A Clustering-Based Optimization Algorithm in Zero-Skew Routings. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 612–616, July 1993.

[25] M. Edahiro and T. Yoshimura. Minimum Path-Length Equi-Distant Routing. In *Proceedings of IEEE Asia-Pacific Conference on Circuits and Systems*, pages 41–46, 1992.

[26] W. C. Elmore. The Transient Response of Damped Linear Networks with Particular Regard to Wide Band Amplifiers. *Journal of Applied Physics*, 19(1):55–63, January 1948.

[27] A. H. Farrahi, C. Chen, A. Srivastava, G. Tellez, and M. Sarrafzadeh. Activity-Driven Clock Design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(6):705 – 714, June 2001.

[28] C. M. Fiduccia and R. M. Mattheyses. A Linear Time Heuristic for Improving Network Partitions. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 175 – 181, 1982.

[29] J.-R. Gao, P.-C. Wu, and T.-C. Wang. A New Global Router for Modern Designs. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 232 – 237, January 2008.

[30] David Garrett, Mircea Stan, and Alvar Dean. Challenges in Clockgating for a Low Power ASIC Methodology. In *International Symposium on Low Power Electronics and Design*, pages 176 – 181, 1999.

[31] L. P. P. P. Ginneken. Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay. In *International Symposium on Circuits and Systems*, pages 865–868, May 1990.

[32] R. T. Hadsell and P. H. Madden. Improved Global Routing through Congestion Estimation. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 28 – 31, June 2003.

[33] C.-H. Hsu, H.-Y. Chen, and Y.-W. Chang. Multi-layer Global Routing Considering Via and Wire Capacities. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 350 – 355, November 2008.

[34] C. P. Hsu. APLS2: A Standard Cell Layout System for Double-layer Metal Technology. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 443 – 448, 1985.

[35] F. K. Hwang, D. S. Richards, and P. Winter. Steiner Tree Problems. *Networks*, 22(1):55 – 89, October 2006.

[36] ISPD. http://vlsicad.eecs.umich.edu/BK/ISPD02bench/. 2002.

[37] ISPD. http://www.sigda.org/ispd2007/rcontest/. 2007.

[38] ISPD. http://www.sigda.org/ispd2008/contests/ispd08rc.html. 2008.

[39] ISPD. http://www.sigda.org/ispd/contests/09/ispd09cts.html. 2009.

[40] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh. Clock Routing for High-Performance ICs. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 573–579, June 1990.

[41] A. Kahng, J. Cong, and G. Robins. High-Performance Clock Routing Based on Recursive Geometric Matching. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 322–327, July 1991.

[42] A. B. Kahng and X. Xu. Accurate Pseudo-Constructive Wirelength and Congestion Estimation. In *Proceedings of International Workshop on System Level Interconnect Prediction*, pages 61 – 68, April 2003.

[43] R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. Vazirani, and V. Vazirani. Global Wire Routing in Two-Dimensional Arrays. *Algorithmica*, 2:113 – 129, 1987.

[44] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh. Pattern Routing: Use and Theory for Increasing Predictability and Avoiding Coupling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(7):777 – 790, July 2002.

[45] B. W. Keinighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell System Technical Journal*, 49:291 – 307, January 1970.

[46] T. Kitahara, F. Minami, T. Ueda, K. Usami, S. Nishio, M. Mruakata, and T. Mitsuhashi. A Clock-Gating Method for Low-Power LSI Design. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 307 – 312, January 1998.

[47] K. Kozminski and E. Kinnen. Rectangular Dual of Planar Graphs. *Networks*, 15:145 – 157, 1985.

[48] B. Krishnamurthy. An Improved Min-Cut Algorithm for Partitioning VLSI Networks. *IEEE Transactions on Computers*, 33:438 – 446, 1984.

[49] Kusnadi and J. D. Carothers. A Method of Measuring Nets Routability for MCMs General Area Routing Problems. In *Proceedings of ACM*

International Symposium on Physical Design, pages 186 – 194, March 1999.

[50] S. T. W. Lai, E. F. Y. Young, and C. C. N. Chu. A New and Efficient Congestion Evaluation Model in Floorplanning: Wire Density Control with Twin Binary Trees. In *European Design and Test Conference*, April 2003.

[51] J. F. Lee and C. K. Wong. A Performance-Aimed Cell Compactor with Automatic Jogs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11:1495 – 1507, 1992.

[52] I.-M. Liu, T.-L. Chou, D. F. Wong, and A. Aziz. Zero-Skew Clock Tree Construction by Simultaneous Routing, Wire Sizing and Buffer Insertion. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 33–38, Nov 2000.

[53] W. H. Liu, Y. L. Li, and H. C. Chen. Minimizing Clock Latency Range in Robust Clock Tree Synthesis. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 389 – 394, January 2010.

[54] J. Lou, S. Krishnamoorthy, and H. S. Sheng. Estimating Routing Congestion Using Probabilistic Analysis. In *Proceedings of ACM International Symposium on Physical Design*, pages 112–117, March 2001.

[55] J. Lu, W. K. Chow, C. W. Sham, and E. F. Y. Young. A Dual-MST Approach for Clock Network Synthesis. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 467 – 473, January 2010.

[56] Y. Luo, J. Yu, J. Yang, and L. Bhuyan. Low Power Network Processor Design Using Clock Gating. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 712 – 715, June 2005.

[57] Y. C. Ma, X. L. Hong, S. Q. Dong, S. Chen, Y. C. Cai, C. K. Cheng, and J. Gu. Dynamic Global Buffer Planning Optimization Based on Detail Block Locating and Congestion Analysis. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 806 – 811, July 2003.

[58] M. D. Moffitt. MaizeRouter: Engineering an Effective Global Router. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 226 – 231, January 2008.

[59] M. D. Moffitt. MaizeRouter: Engineering an Effective Global Router. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(11):2017 – 2026, November 2008.

[60] M. D. Moffitt, J. A. Roy, and I. L. Markov. The Coming of Age of (Academic) Global Routing. In *Proceedings of ACM International Symposium on Physical Design*, pages 148 – 155, April 2008.

[61] G.-J. Nam, C. Sze, and M. Yildiz. The ISPD Global Routing Benchmark Suite. In *Proceedings of ACM International Symposium on Physical Design*, pages 156 – 159, April 2008.

[62] G.-J. Nam, M. Yildiz, D. Z. Pan, and P. H. Madden. ISPD Placement Contest Updates and ISPD 2007 Global Routing Contest. In *Proceedings of ACM International Symposium on Physical Design*, pages 167 – 167, April 2007.

[63] S. Natarajan, S. L. Sam, D. Boning, A. Chandrakasan, R. Vallishayee, and S. Nassif. A Methodology for Modeling the Effects of Systematic Within-Die Interconnect and Device Variation on Circuit Performance. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 172–175, June 2000.

[64] J. Oh and M. Pedram. Gated Clock Routing Minimizing the Switched Capacitance. In *European Design and Test Conference*, pages 692 – 697, 1998.

[65] J. Oh and M. Pedram. Gated Clock Routing for Low-Power Microprocessor Design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(6):715 – 722, June 2001.

[66] T. Ohtsuki, T. Sudo, and S. Goto. CAD Systems for VLSI in Japan. *Information and Control*, 59, 1983.

[67] R. H. J. M. Otten. Efficient Floorplan Optimization. In *Proceedings of IEEE International Conference on Computer Design*, pages 499 – 503, 1983.

[68] M. M. Ozdal and M. D. F. Wong. ARCHER: A History-driven Global Routing Algorithm. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 481 – 487, November 2007.

[69] M. Pan and C. Chu. FastRoute: a Step to Integrate Global Routing into Placement. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 464 – 471, November 2006.

[70] M. Pan and C. Chu. FastRoute 2.0: A High-quality and Efficient Global Router. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 250 – 255, January 2007.

[71] A. Rajaram, J. Hu, and R. Mahapatra. Reducing Clock Skew Variability via Crosslinks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1176–1182, June 2006.

[72] A. Rajaram and D. Z. Pan. Variation Tolerant Buffered Clock Network Synthesis with Cross Links. In *Proceedings of ACM International Symposium on Physical Design*, pages 157–164, April 2006.

[73] S. Rao. Finding Near Optimal Separators in Planar Graphs. In *Annual Symposium on Foundations of Computer Science*, pages 225 – 237, 1987.

[74] J. A. Roy and I. I. Markov. High-performance Routing at the Nanometer Scale. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 496 – 502, November 2007.

[75] M. Saeedi, M. S. Zamani, and A. Jahanian. Prediction and Reduction of Routing Congestion. In *Proceedings of International Workshop on System Level Interconnect Prediction*, pages 72 – 77, April 2006.

[76] R. Saleh, S. Z. Hussain, S. Rochel, and D. Overhauser. Clock Skew Verification in the Presence of IR-Drop in the Power Distribution Network. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(6):635 – 644, June 2000.

[77] L. A. Sanchis. Multi-Way Network Partitioning. *IEEE Transactions on Computers*, 38:62 – 81, 1989.

[78] A. Sangiovanni-Vincentelli and M. Santomauro. YACR: Yet Another Channel Router. In *Proceedings of Custom Integrated Circuits Conference*, pages 460 – 466, 1982.

[79] M. Sarrafzadeh and C. K. Wong. An Introduction to VLSI Physical Design. In *The McGraw-Hill Companies, Inc.*, 1996.

[80] S. Sauter, D. Schmitt-Landsiedel, R. Thewes, and W. Webber. Effect of Parameter Variations at Chip and Wafer Level on Clock Skews. *IEEE Transactions on Semiconductor Manufacturing*, 13(4):395–400, November 2000.

[81] D. G. Schweikert and B. W. Kernighan. A Proper Model for the Partitioning of Electrical Circuits. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 57 – 62, 1972.

[82] C. Sechen and A. Sangiovanni-Vincentelli. TimberWolf3.2: A New Standard Cell Placement and Global Routing Package. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 432 – 439, 1986.

[83] C. W. Sham and E. F. Y. Young. Congestion Prediction in Early Stages. In *Proceedings of International Workshop on System Level Interconnect Prediction*, pages 91–98, 2005.

[84] C. W. Sham and E. F. Y. Young. Congestion Prediction in Floorplanning. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 1107–1110, June 2005.

[85] E. Shargowitz and J. Keel. A Global Router Based on Multicommodity Flow Model. *Integration: The VLSI Journal*, 5:3 – 16, 1987.

[86] X. W. Shih, C. C. Cheng, Y. K. Ho, and Y. W. Chang. Blockage-Avoiding Buffered Clock-Tree Synthesis for Clock Latency-Range and Skew Minimization. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 395 – 400, January 2010.

[87] L. Stockmeyer. Optimal Orientation of Cells in Slicing Floorplan Designs. *Information and Control*, 57:91 – 101, 1983.

[88] C. N. Sze, P. Restle, G.-J. Nam, and C. Alpert. ISPD2009 Clock Network Synthesis Contest. In *Proceedings of ACM International Symposium on Physical Design*, pages 149–150, March 2009.

[89] T. Taghavi, F. Dabiri, A. Nahapetian, and M. Sarrafzadeh. Tutorial on Congestion Prediction. In *Proceedings of International Workshop on System Level Interconnect Prediction*, pages 15 – 24, April 2007.

[90] H. Tang and W. K. Chen. Generation of Rectangular Duals of a Planar Triangulated Graph by Elementary Transformations. In *International Symposium on Circuits and Systems*, pages 2857 – 2860, 1989.

[91] G. E. Tellez, A. Farrahi, and M. Sarrafzadeh. Activity-Driven Clock Design for Low Power Circuits. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 62 – 65, November 1995.

[92] R.-S. Tsay. Exact Zero Skew. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 336–339, Nov 1991.

[93] S. Tsukiyama, K. Tani, and T. Maruyama. A Condition for a Maximal Planar Graph to Have a Unique Rectangular Dual and Its Application to VLSI Floor-plan. In *International Symposium on Circuits and Systems*, pages 931 – 934, 1989.

[94] M. Wang and M. Sarrafzadeh. Modeling and Minimization of Routing Congestion. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 185 – 190, July 2000.

[95] M. Wang, X. Yang, and M. Sarrafzadeh. Congestion Minimization During Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(10):1140 – 1148, October 2000.

[96] J. Westra, C. Bartels, and P. Groeneveld. Probabilistic Congestion Prediction. In *Proceedings of ACM International Symposium on Physical Design*, pages 204–209, March 2004.

[97] D. F. Wong and C. L. Liu. Floorplan Design of VLSI Circuits. *Algorithmica*, 4:263 – 291, 1989.

[98] L. Xiao, L. Li, and Z. Qian. AMGR in International Symposium on Physical Design. April 2008.

[99] X. J. Yang, R. Kastner, and M. Sarrafzadeh. Congestion Estimation During Top-Down Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(1):72 – 80, January 2002.

[100] S. Zanella, A. Nardi, A. Neviani, M. Quarantelli, S. Saxena, and C. Guardiani. Analysis of the Impact of Process Variations on Clock Skew. *IEEE Transactions on Semiconductor Manufacturing*, 13(4):401 – 407, November 2000.

[101] Y. Zhang, Y. Xu, and C. Chu. FastRoute 3.0: A Fast and High Quality Global Router Based on Virtual Capacity. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pages 344 – 349, November 2008.