ANALYTICAL LAYER PLANNING FOR NANOMETER VLSI DESIGNS

A Thesis

by

CHI-YU CHANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2012

Major Subject: Electrical Engineering

ANALYTICAL LAYER PLANNING FOR NANOMETER VLSI DESIGNS

A Thesis

by

CHI-YU CHANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Jiang Hu
Committee Members,	Weiping Shi
	Rabi N. Mahapatra
Head of Department,	Costas Georghiades

August 2012

Major Subject: Electrical Engineering

ABSTRACT

Analytical Layer Planning for Nanometer VLSI Designs.

(August 2012)

Chi-Yu Chang, B.S., National Chung Cheng University, Taiwan

Chair of Advisory Committee: Dr. Jiang Hu

In this thesis, we proposed an intermediate sub-process between placement and routing stage in physical design. The algorithm is for generating layer guidance for post-placement optimization technique especially buffer insertion. This issue becomes critical in nowadays VLSI chip design due to the factor of timing, congestion, and increasingly non-uniform parasitic among different metal layers. Besides, as a step before routing, this layer planning algorithm accounts for routability by considering minimized overlap area between different nets. Moreover, layer directive information which is a crucial concern in industrial design is also considered in the algorithm.

The core problem is formulated as nonlinear programming problem which is composed of objective function and constraints. The problem is further solved by conjugate gradient method. The whole algorithm is implemented by C++ under Linux operating system and tested on ISPD2008 Global Routing Contest Benchmarks. The experiment results are shown in the end of this thesis and confirm the effectiveness of our approach especially in routability aspect.

To my family

ACKNOWLEDGEMENTS

I would like to thank all those who encouraged me and helped me during my study and research at Texas A&M University.

Especially, I would like to thank my family for their ceaseless support, encouragement and endless love. Without which I would never able to complete my master degree so smoothly.

Also, I would like to extend my heartfelt gratitude to my advisor, Dr. Jiang Hu, who gave me constant guidance as well as warm encouragement throughout this research project. He was always patient, kind and helpful whenever I had questions on my academic life. I could not have completed this thesis and knew physical design this wonder world without his guidance and generous support. I also would like to thank Dr. Weiping Shi and Dr. Rabi Mahapatra for being my committee members, and for their suggestions on this research.

Last but not least, thanks to all my friends and colleagues who accompany with me these years, and also the department faculty and staff for giving me a warm and kind environment during my life at Texas A&M University.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

1. INTRODUCTION

1.1. Post-placement Optimization in VLSI Designs

Traditionally, physical design produces the geometrical data like netlist of the circuits for fabrication, and the process is mainly partitioned into two parts: placement and routing. In typical design flows, many optimizations are performed between the placement and routing, such as gate sizing and buffer insertion. Our work is to generate layer guidance especially for pre-routing optimization, especially buffer insertion. In modern technology, this issue became critical because the number of metal layers keeps increasing and metal size/parasitic among different layers becomes increasingly non-uniform. As such, without layer and corresponding parasitic information, the post-placement (pre-routing) optimizations may unnecessarily insert huge amount of buffers. Therefore, it is essentially important to provide a reliable estimation on layer information for the optimizations.

1.2. Routability-driven Techniques

Steps before routing like placement and buffer insertion need to consider routability so that they can lead to routable circuits. Therefore, routability-driven techniques have been proposed and broadly used in many designs as in [1-4]. Being a pre-routing process

---

This thesis follows the style of *IEEE Transactions on Computer Aided Design of Integrated Circuits and System*.

as well in this design, we will take it into consideration.

1.3. Layer Directive Information

Besides, in order to boost the development of routing techniques, International Symposium on Physical Design (ISPD) released two sets of benchmarks in 2007 and 2008 respectively. Both sets provide the multi-layer designs; as a result, recent developed routing algorithm as in [5] and [8] could generate 3D routing solution with minimized wirelength and overflow considered.

However, in industrial design, the real objectives of router are not only considering wirelength and overflow but also the detours of those timing-critical nets into higher layer. Layer directives information came out for those critical nets in ICCAD 2009 benchmark [9], and it could be considered by router [6] and [7] so as to meet the system timing specifications.

1.4. Objectives

Our objective is to develop a pre-routing layer planner, which can efficiently generate guidance for further buffer insertion. This planner assigns each net to a layer range such that the estimated wire congestion is minimized and the layer directives for timing critical nets are observed. Since this is prior to routing, a net is represented by a netbox, which is the smallest bounding rectangle covering all pins of the net. A netbox is illustrated in Figure 1.

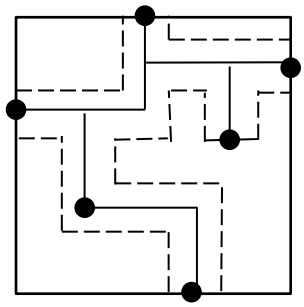


Fig. 1. Rectangle is called netbox or net and each connecting wire between pins is segment.

1.5. Thesis Organized

The reminder of this thesis is organized as follows. Section2 is to introduce the previous similar work. Section3 formulates the 3D layer planning problem and mentions the method how we solve the problem. Section4 introduces our analytical layer planner engine explicitly and give formal explanation on each parameter we used in this work. Senction5 give the detailed implementation on solving the formula by conjugate-gradient nonlinear programming. Finally, we will have experiment results and comparison charts in Section6 and Section7 respectively, and the thesis would conclude in Section8.

2. PREVIOUS WORK

A straightforward way to implement layer planning is to run a global router [5-7], stop it pre-maturely, and then take the result as layer guidance. The layer planning results will serve as a soft guidance, instead of hard constraints, to subsequent routers. This method seems right and reasonable, and we will rename this pre-mature NTHU-route as NTHU-layer planner. However, we innovate a new method to implement it; no like mentioned NTHU-route, their layer assignment algorithm is to assign each segment to different layer as in [10]. What we do is to assign whole netbox into discrepant layer. In theory, the problem spaces in our work would be smaller compared to the method in [10], since one netbox includes a bunch of segments. This attribute also meet the needs of our design, since we are pursuing a briefly guidance algorithm which should have the ability to generate sound information efficiently.

To carefully rectify the terms we are using, as layer assignment [10] in NTHU-route is doing some similar work with us but focusing on different aspect, we would use the term: layer planner to represent our work from now on.

As mentioned in Section 1, proposed work will finally be inserted to provide guidance for buffer insertion, and then to global router. The result of proposed work should not only generate guidance lead to less buffer using but also better routing ability.

By measuring routability, we will check firstly if the routing attributes in terms of wirelength, via count, overflow and routing time, become better or not compared to

purely exploiting NTHU-route. Furthermore, we will also do routing attributes comparison between what NTHU-layer planner and our layer planner provide. Flow for above comparison is shown in Fig. 2.

As for judgement of buffer insertion ability in our design, we would apply our layer guidance into buffer insertion algorithm to see if less buffer inserted and smaller buffer area occupied compared to which without any layer guidance. However, the buffer insertion algorithm we preferred still in experimental stage, we would combine both work in the future to see if it provide efficient and sound buffer insertion ability. One thing to note is that, in the buffer insertion stage, we will need to provide physical implementation details such as layer parasitic information, net's RAT (Required Arrival Time) and wire width of each and every different layers to get the realistic timing constraint. Since adding buffer is to tackle the issue of timing violation in real design.

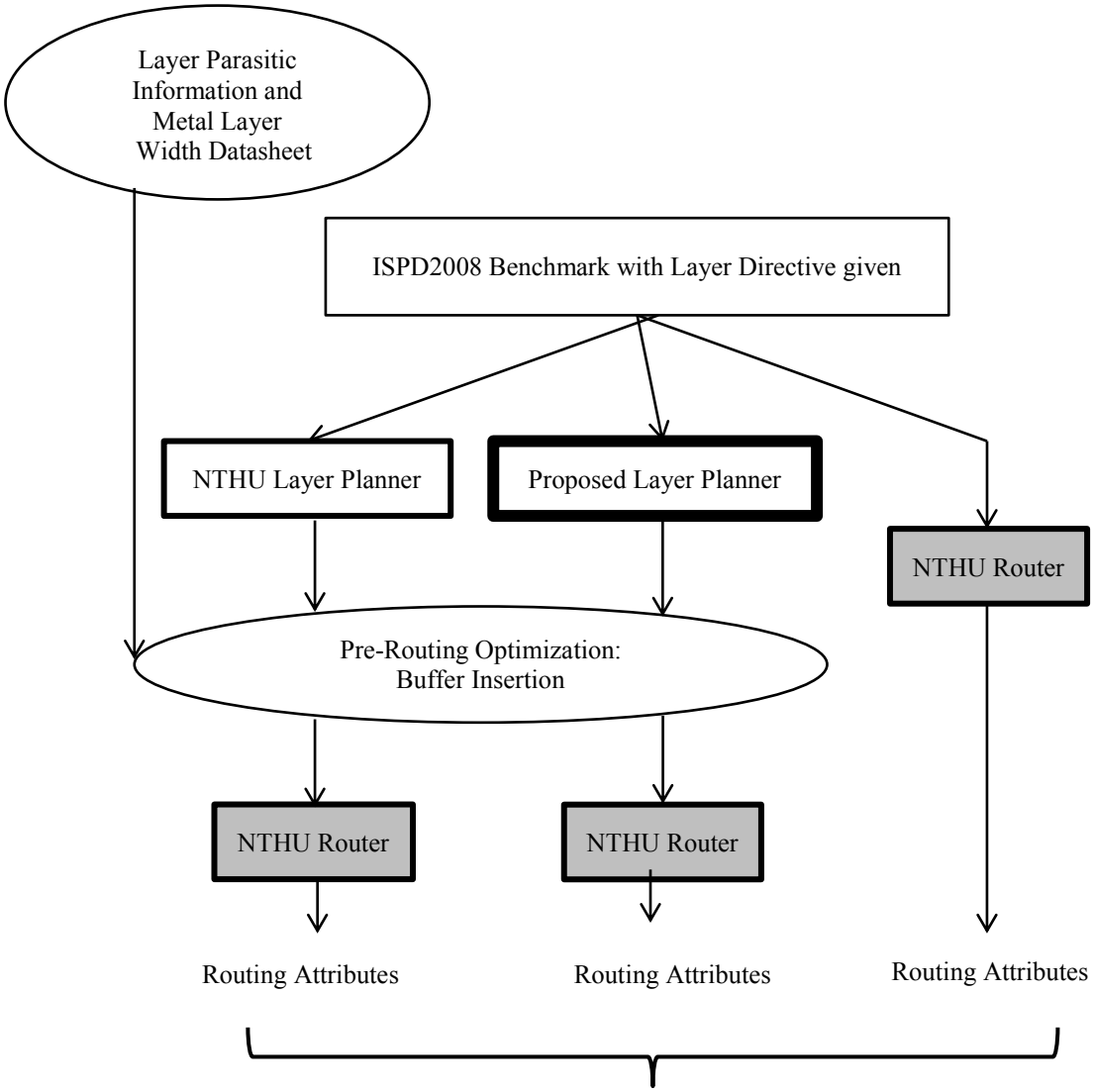


Fig. 2. Flow for final result comparison

3. PROBLEM FORMULATION

What we want to fulfill is giving each netbox a layer range to lie in. The reason for saying layer range rather than a specific layer is due to nowadays routing technique will have the wire routed in interleaving direction in consecutive layer. So we can regard consecutive two layers as one set of routing resource and call it tier. Fig.3 is to show this concept. In this two layer resource example, layer2 get wider wire and goes in vertical direction; while, layer1's wire are thinner and aligned in horizontal way.

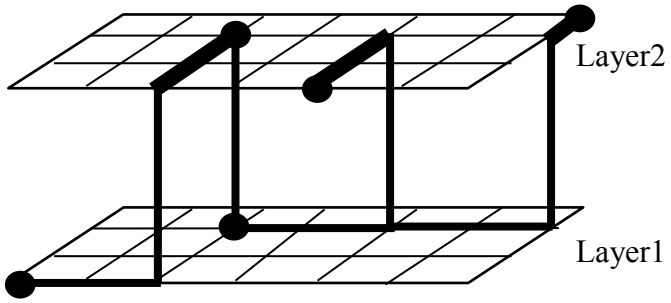


Fig. 3(a). Wire routed in interleaving direction in consecutive layer

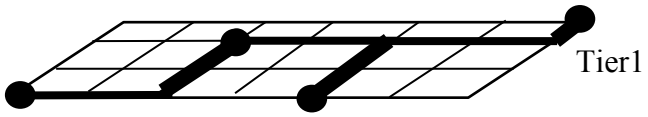


Fig. 3(b). Regard consecutive two layers as one set of routing resource

Besides giving guidance for netbox a layer range to lie in, we will also take routability and layer directive suggestion from benchmark into account for further design. Minimizing the overlap area between each netbox can help relief the congestion issue

and make routing easier. However, the given benchmark is well-placed, so we won't make any change on $\boldsymbol{x-y}$ coordinate of netboxes. Hence, the routability issue would become minimizing the netbox overlap area by moving them into different tier in $\boldsymbol{z}$ direction.

Given a circuit represented as hypergraph $\boldsymbol{H=(V,E)}$, the placement region $\boldsymbol{R}$ and the total number of device tiers $\boldsymbol{K}$; where $\boldsymbol{V}$ represent a set of netboxes and $\boldsymbol{E}$ for the coordinate $(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \boldsymbol{R}$ of those given well-placed $\boldsymbol{i}$ netboxes. The task is to assign every netbox $\boldsymbol{v_i} \in \boldsymbol{V}$ a $\boldsymbol{z_i}$ value, which indicates that this netbox is placed onto the tier $\boldsymbol{z_i} \in \{\mathbf{1}, \mathbf{2}, \cdots, \boldsymbol{K}\}$, so as to minimize the netboxes overlap area. To get routable result, we have to include a constraint that $\boldsymbol{z_i}$ has to be a number between available layer resource ranges. Finally, we formulate the routability-driven layer planner problem as overlap area minimization in (1).

$$\begin{cases} \boldsymbol{Objective\ Function} = \sum\left(\boldsymbol{netbox_{overlap}}\right) \\ \boldsymbol{s.t.}\ \ \mathbf{0} < \boldsymbol{z_i} < \boldsymbol{K}+\mathbf{1} \end{cases} \tag{1}$$

Beside the overlap, consideration on timing critical issue will be implemented by adding a BETA factor in the objective function. BETA would be set to smaller if being put within layer range suggested by layer directive information.

Another RUDY factor will also in our discussion by taking practical design into account. RUDY will facilitate netbox to go to lower layer due to plenty routing resource in normal cases; otherwise, it is necessary to route in higher layer for getting smaller

overlap or better timing alignment. The issue will become making decision by observing tradeoff among these factors – overlap area, BETA and RUDY factor value. We will address more details on how we derive these factors in Section4.

After we figure out all the factors in objective function, we employed optimization method – conjugate gradient to handle the 3D layer planning, which will also be explicitly explained in Section5. Moreover, the solution space of nonlinear programming is Real number $\boldsymbol{R}$, our discussion on layer planner is always Natural number $\boldsymbol{N}$. Thus, we need a mechanism to do relaxation from discrete number to continuous, which will be explicitly shown in Section4.2. The overall layer planner flow is shown in Fig. 4.

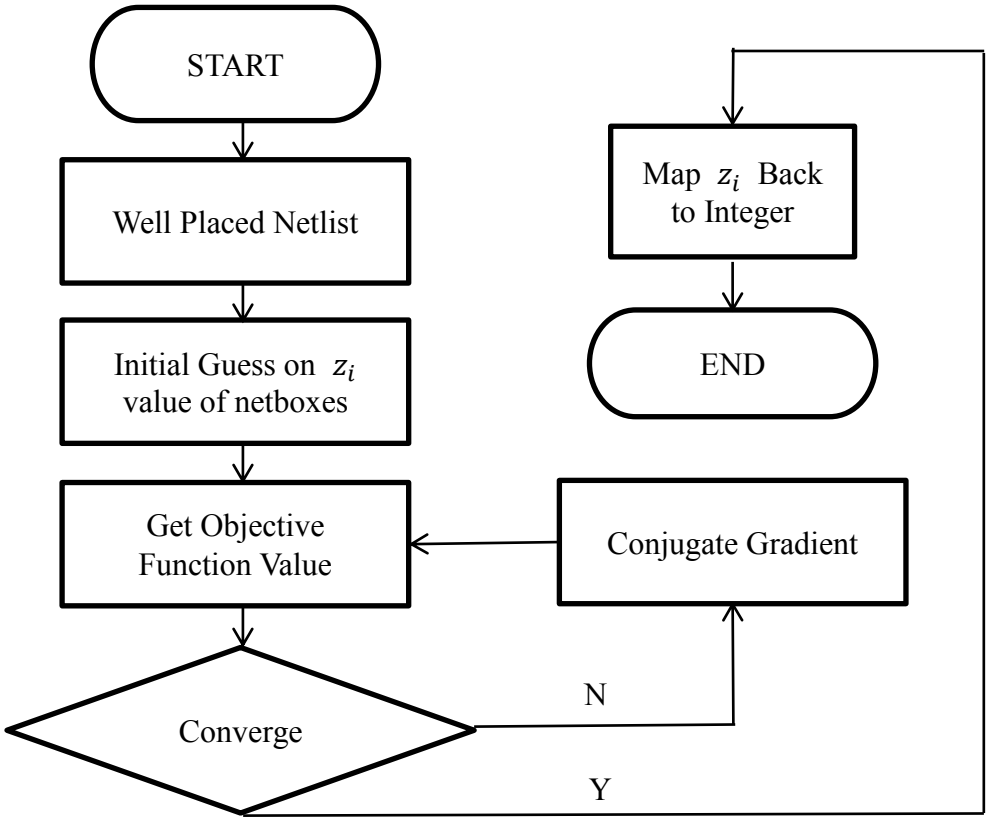


Fig. 4. Proposed layer planner algorithm flow

4. ANALYTICAL LAYER PLANNER ENGINE

According to (1), we could transform the desired problem by setting up boundary for $\boldsymbol{z_i}$ when doing the iteration in nonlinear programming. Besides, include the factors mentioned in above chapter, we can get complete objective function in (2), and those $\boldsymbol{z_i}$ value to make objective function minimized would be our desired layer planning solution.

$$\boldsymbol{Objective\ Function} = \sum \begin{pmatrix} \boldsymbol{Netbox_{overlap}}(\boldsymbol{z_i}) \times \boldsymbol{Relaxation}(\boldsymbol{z_i}) \\ \times \boldsymbol{RUDY}(\boldsymbol{z_i}) \times \boldsymbol{BETA}(\boldsymbol{z_i}) \end{pmatrix}\Big|_{\boldsymbol{z_i} \in [\boldsymbol{1}, \cdots, \boldsymbol{K}]} \quad (\boldsymbol{2})$$

4.1. Density Penalty Function

In modern analytical placer [11-14], the overlap-free issue are broadly studied and usually transformed to density penalties function as in [12, 13]. We will use this concept in formulation so as to find out the minimized overlap area between each netboxes. The density penalty function is for minimizing the overlap by moving netboxes in z-direction and the minimized value should lead to the least overlapping area.

Assume that every netbox has a legal tier assignment ($\boldsymbol{i.e.}\ \ \boldsymbol{z_i} \in \{\boldsymbol{1}, \boldsymbol{2}, \cdots, \boldsymbol{K}\}$), then we can define $\boldsymbol{K}$ different layer density penalty function for each of these $\boldsymbol{K}$ tiers and problem would be simplified as summation of all these layer density penalty function one by one.

To further increase the calculation efficiency; and after all, no much loss on precision, we will generate overlap area based on netbox and gcell, which is a placement region ***R*** divided into many square tiles as illustrated in Fig. 5., rather than overlap between different netbox.

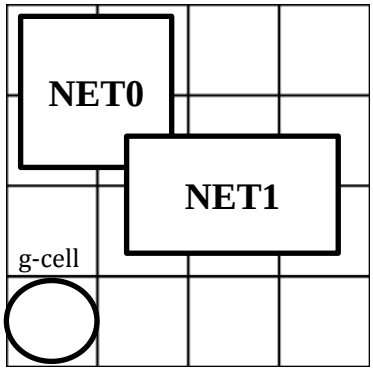


Fig. 5. Illustration on g-cell

By observing the benchmark setting, we can conclude that calculation on overlap between netbox and g-cell won't have too much loss on precision compared to real overlap area between each netboxes we pursued. This is due to the g-cell dimension stated in benchmarks are comparably tiny when it comes to netbox dimension. Fig.6. is to show the explanation for this argument. Two identical placement region ***R*** are shown in Fig. 6(a) and Fig. 6(b), as two identical netboxes place on it but with different g-cell dimension. Fig. 6(a)'s g-cell density is nine times smaller than Fig. 6(b) and we can find out in this sparse g-cell, there is overlap area miscalculation occurred. Since, theoretically, netbox1 and netbox2 don't have overlap at all in this example; however, by using proposed estimation on overlap area between netbox and g-cell, g-cell_3 in Fig. 6(a) will make things disobey from truth. That is, in our calculation, situation in Fig. 6(a)

will both have overlap on g-cell_3 and make it seems to have overlap. From observing, we can notice that denser g-cell like Fig. 6(b) will relief this miscalculation situation after all, since these two netboxes don't have overlap area on same g-cell.

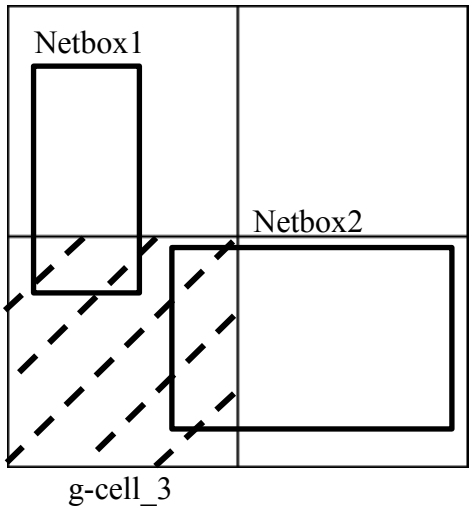


Fig. 6(a). Sparse g-cell will cause miscalculation on overlap area in g-cell 3

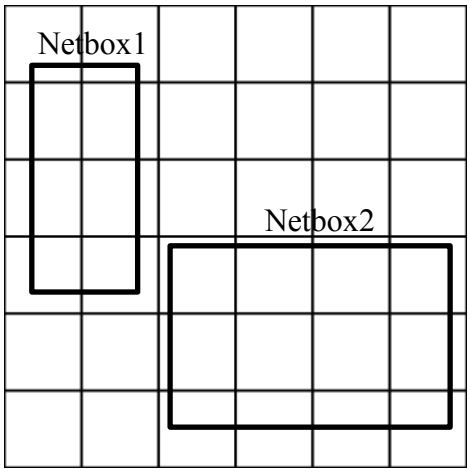


Fig. 6(b). Dense g-cell dimension will relief miscalculation, since there is no overlap in a specific g-cell

Inspired by the quadratic penalty terms in 2D placement methods [15] and observing the following cases, we can have rather precise formula to calculate the netbox and g-cell overlap. We use simple 2D plane example in the following to explain it and the proposed 3D method in this thesis is based on it. Suppose given a 2D plane, there are 4 netboxes but only three tiers available; and netbox0(n0) is fixed on tier one, then the scattering of these four nteboxes could be categorized into 3 cases. The Fig. 7 shows these three cases and has $\boldsymbol{z}$ direction in vertical as different tier to place, and $\boldsymbol{x}$

direction in horizontal as different g-cell. Each rectangle represent one netbox in cross-section view and each with two unit length and one unit height; besides, length unit is like how many g-cell it cross and height unit is like the area a netbox occupied in that specific g-cell. Among these three cases, we can easily observe that overlap the most happened in case three; and by the quadratic term, we can also get the value the biggest. Thus, quadratic term is used as punishment when overlap happened.

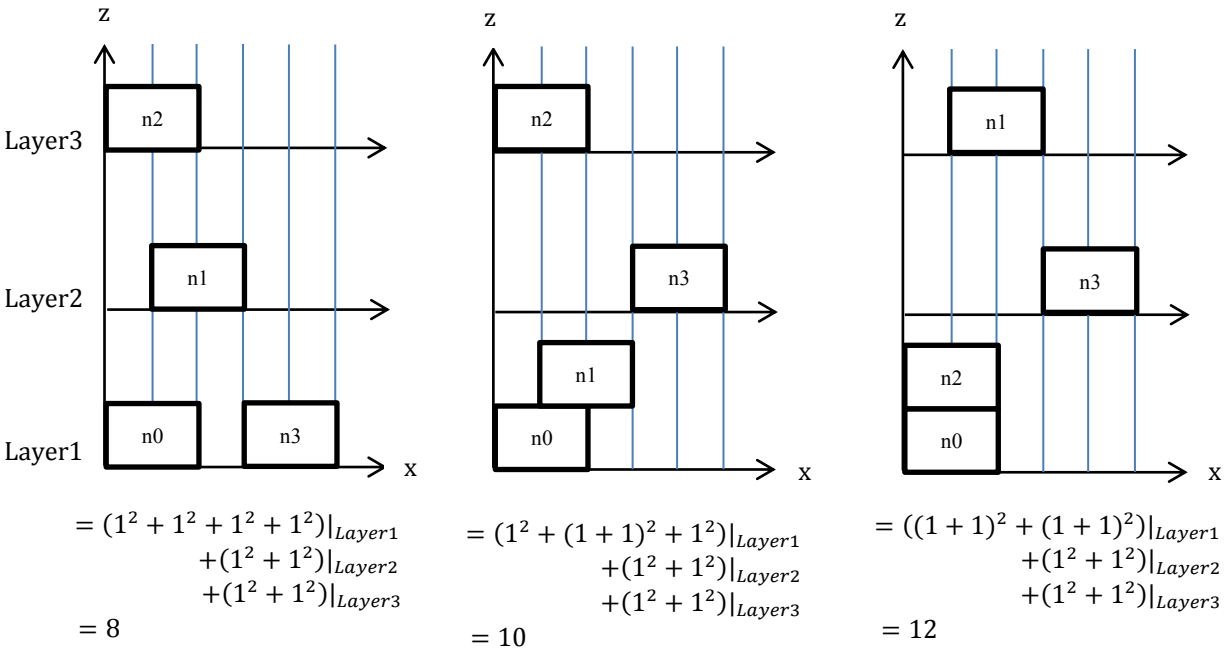


Fig. 7. Three possible cases to demonstrate the necessity of quadratic term.

Conclude from what we discussed above, we can now have the prototype of our density penalty function, which is shown in (3), to emulate netbox and g-cell overlap area calculation.

$$\boldsymbol{DensityPenalty}(\boldsymbol{z}) = \sum_{\boldsymbol{layer}} \left\{ \sum_{\boldsymbol{g-cell}} \left[\sum_{\boldsymbol{netbox}} (\boldsymbol{netbox\_gcell\_overlap}) \right]^2 \right\} \qquad (3)$$

4.2. Relaxation of Discrete Variables

As mentioned in Section3. The layer planning variables are represented by z_i, which should be a discrete variable in $\{1,2,\cdots,K\}$. However, the range of z_i have to be relaxed from the discrete number set $\{1,2,\cdots,K\}$ to continuous interval $[1,K]$, so that we can assert nonlinear solver in our layer planner engine. Moreover, in order to get legal integer solution, the relaxed continuous results after optimization have to be mapped back to the most closed discrete value at the end. We borrow the idea from bell-shaped function $\eta(layer,z_i)$ as in [1, 4, 12, 16] to help us doing relaxation by projecting those non-integer placed netbox to integer layer. Equation(4) shows the proposed method, where $\boldsymbol{z}$ represent the netbox placed layer and $\boldsymbol{k}$ is device integer tier.

$$\boldsymbol{\eta(layerk,z)|_{1\le k\le K}} = \begin{cases} \boldsymbol{1-2(z-k)^2} & \boldsymbol{,\ |z-k|\le 1/2} \\ \boldsymbol{2(|z-k|-1)^2} & \boldsymbol{,\ 1/2<|z-k|\le 1} \\ \boldsymbol{0} & \boldsymbol{,\ otherwise} \end{cases} \tag{4}$$

An actual curve of how this relaxation works for a resource with three layers is given in Fig. 8(a). Three curves noted by Layer1, Layer2 and Layer3 represents tier 1, 2 and 3 respectively. The x-axis is the relaxed layer number in z-direction, while the y-axis indicates the amount of function value to be projected in the actual discrete tiers. An example is shown in Fig. 8(b), a netbox is temporarily placed at $z_i = 1.2$ between device tier 1 and 2. The relaxation function will project 92% of its original function

value onto tier1 and 8% to tier2. By this method, we can have our mapping from a continuous number, which is used in nonlinear programming, to discrete number for sigma numerical calculation in objective function as in (3).

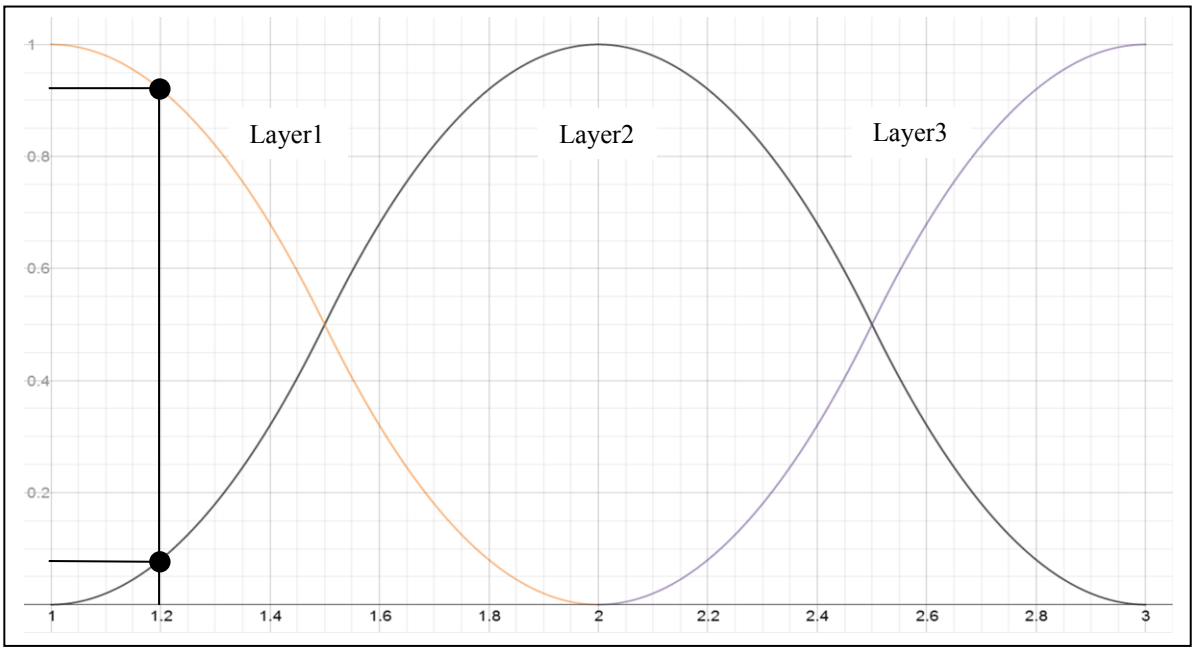


Fig. 8(a). Function $\boldsymbol{\eta(layerk, z)}$ help implement relaxation

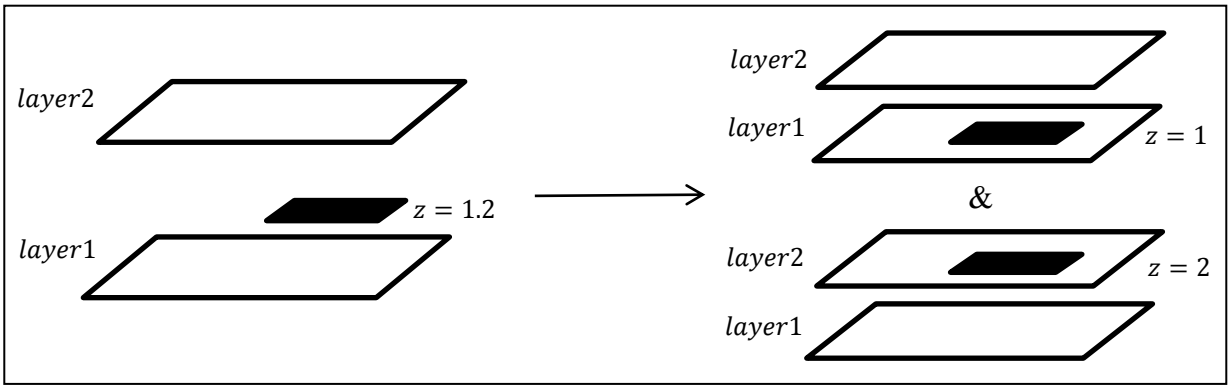


Fig. 8(b). A netbox with value 1 placed at $\boldsymbol{z_i = 1.2}$
will project 0.92 onto tier 1 and 0.08 onto tier 2

The relaxation function establish a continuous to discrete number transfer mechanism; however, we should also let z_i to be located the closer to integer the better, so that we can have more precise and less estimation solution, since we will map back nonlinear programming solution to the nearest device tier at the end. In order to avoid the situation happened in Fig. 9, which is caused by original relaxation function: right figure even have smaller objective value than the left one, which is theoretically the best tier to put in this small example. The defect for the relaxation function is when some netbox being put onto integer layer, it will give complete one projection ratio on that layer; however, when one netbox located at non-integer number tier, by the relaxation function and quadratic term we mentioned in Section4.1, it will make total summation of projection ratio on adjacent integer layer smaller than one. Fig. 9 shows this defect on equation expression.

We will then introduce the interlayer relaxation function in (5). By adding this function, it help getting more integer-closed results because the tendencies of reaching

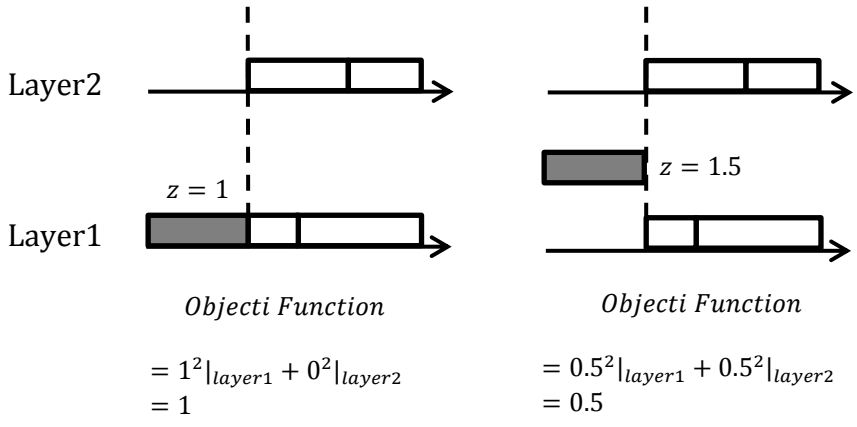


Fig. 9. Showing defect of only relexation function

smaller objective function. Fig.10 is improved by adding interlayer relaxation function.

$$\boldsymbol{\eta\_interlayer(layerk,z)|_{1\le k\le K-1}} = \begin{cases} \mathbf{1-2(z-k+0.5)^2}\ , & \mathbf{|z-k+0.5|\le 1/2} \\ \mathbf{2(|z-k+0.5|-1)^2}\ , & \mathbf{1/2<|z-k+0.5|\le 1} \\ \mathbf{0}\ , & \boldsymbol{otherwise} \end{cases} \quad (5)$$

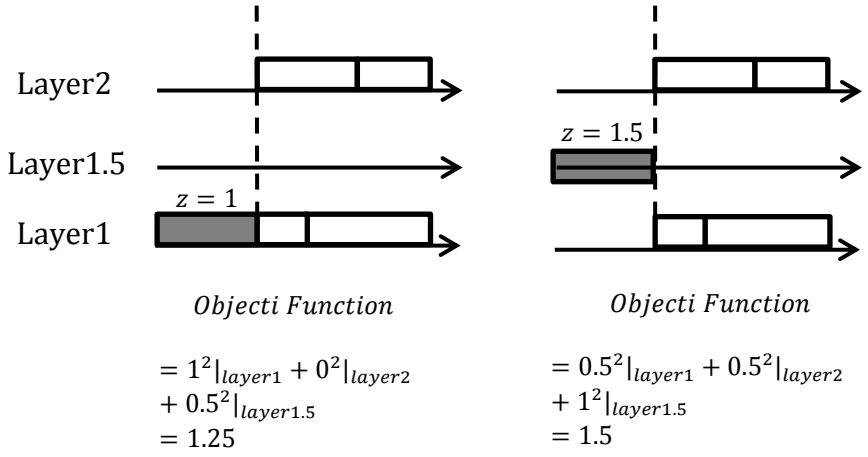


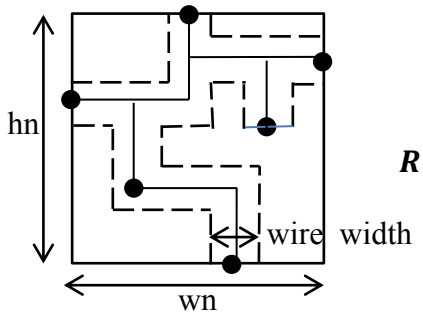
Fig. 10. Improved by adding interlayer relaxation function

4.3. Estimation of the Routing Demand

Because routability is an abstract concept, it is hard to tell whether a process can be routed easily or not. Therefore, we have to apply some metric to measure the routability. Many previous works are to evaluate the routability as in [17]; and here, we generalize the routability evaluation technique called RUDY [2]. RUDY stands for Rectangular Uniform wire DensitY and is defined as the ratio of the wire area WA_n to the netbox

area NA_n, which is shown in Fig. 11. However, wire area cannot be known until the routing stage finished, so RUDY will use Rectilinear Steiner Minimal Tree (RSMT) to estimate routing model.

Furthermore, RUDY value not only shows the congestion but also give us different penalty when we put netbox in different layer. Generally speaking, putting netbox on lower layer is preferred, since more routing resource due to smaller wire width; however, when it comes to too much overlap in lower layer level or layer directive information are included, which will make overall objective function too big, the located layer would be detoured into higher layer by nonlinear programming decision. As a result, adding RUDY factor in objective function will give us more realistic solution in terms of considering difference wire width in discrepant device layer.



$$\boldsymbol{RUDY} = \frac{\boldsymbol{Wire\_Area}}{\boldsymbol{Net\_Area}} = \frac{\boldsymbol{wire\_width(z)} \times (\boldsymbol{w_n} + \boldsymbol{h_n})}{\boldsymbol{w_n} \times \boldsymbol{h_n}}$$

Fig. 11. RUDY function and its representation

4.4. BETA Parameter for Timing Critical Nets

Start from ICCAD 2009 [9], some benchmark will include layer directive information for timing critical nets. This information shows these nets have tighter slack

and should be put in higher layer, where wider wire will make signal conduct faster so as to release the timing violation situation. Depending on different timing request, some may ask netbox goes to the highest layer; others only need not to be located at the lowest layer. However, layer directive information is not hard constraint in this work. That is, there is no need to strictly follow the layer directive information when we do layer planning.

Beta is a parameter for us to punish objective function on netbox if it is not following the suggestion of layer directive from benchmark. In each iteration, if the netboxes went to a layer happened to be in the range of given layer directive, then we will insert smaller Beta; otherwise, the bigger Beta value would be asserted. Hence, theoretically, every timing-critical netbox would try to locate at suggested layer to keep the objective function minimized; however, we regard timing critical netbox's layer directive as a soft constraint as we mentioned above, so there is no need to strictly stick to it. The solutions are finally decided in nonlinear programming method to reach minimum objective function by weighing the tradeoff between these factors.

Conclude from Section4, we can generate our complete formula in (6).

$$\begin{aligned}\boldsymbol{Objective\ function} = &\sum_{\boldsymbol{Layer}}\left\{\sum_{\boldsymbol{g-cell}}\left[\sum_{\boldsymbol{netbox}}\left(\begin{matrix}\boldsymbol{Netbox_{overlap}(z_i)\times\eta(z_i)}\\ \boldsymbol{\times RUDY(z_i)\times BETA(z_i)}\end{matrix}\right)\Big|_{\boldsymbol{z_i\in[1,\cdots,K]}}\right]^{\boldsymbol{2}}\right\}\\ &+\sum_{\boldsymbol{Layer}}\left\{\sum_{\boldsymbol{g-cell}}\left[\sum_{\boldsymbol{netbox}}\left(\begin{matrix}\boldsymbol{Netbox_{overlap}(z_i)\times\eta_{interlayer(z_i)}}\\ \boldsymbol{\times RUDY(z_i)\times BETA(z_i)}\end{matrix}\right)\Big|_{\boldsymbol{z_i\in[1,\cdots,K]}}\right]^{\boldsymbol{2}}\right\}\end{aligned} \quad (\boldsymbol{6})$$

5. DETAIL IMPLEMENTATION

We use the conjugate gradient method to minimize the objective function as described in (6). A detailed treatment along with explicitly explanation on conjugate gradient method could be found in [18-20]. Generally speaking, the conjugate gradient method finds the optimum value by executing a series of line search. By doing that, we have to know where the right direction to go and how big a step is at a time, and that is called search direction and step length respectively. The result of one line search is used as the start point for the next line search and we won't stop the iteration until customized criteria reached. The line search has the following form as (7).

$$\boldsymbol{z_{i+1} = z_i + \alpha_i \cdot d_i}$$

$$\begin{cases} \boldsymbol{d_i = -g_i + \beta_i \cdot d_{i-1}} & \boldsymbol{for\ i > 1} \\ \boldsymbol{d_i = -g_i} & \boldsymbol{for\ i = 1} \end{cases} \tag{7}$$

Where g_i denote the gradient, α_i represent step length derived from Newton-Raphson algorithm and d_i is the search direction with the using of Fletcher-Reeves formula on β. For step length α_i and factor for search direction β, there's a large variety of choices we can have, but here we just take general used one Newton-Raphson and Fletcher-Reeves respectively. As for the stop point, we will not halt conjugate gradient iteration until the following criteria are reached: 1. Predetermined number of iteration has hit; 2. the function value is not changing

significantly with additional iterations. The complete implementation flow of the conjugate gradient method is shown in Fig. 12.

Given a function $\boldsymbol{f}$, its negative gradient value $\boldsymbol{r_i}$, hessian value $\boldsymbol{h_i}$, an starting value vector $\boldsymbol{z_i}$, which store the initial located tier for each and every netbox. A maximum number of iteration $\boldsymbol{iter_{max}}$, an error tolerance $\boldsymbol{\varepsilon} < 1$, a maximum number of Newton-Raphson iteration $\boldsymbol{j_{max}}$ and a Newton-Raphson error tolerance $\boldsymbol{\epsilon}$. This conjugate gradient algorithm will terminate when the maximum number of iteration $\boldsymbol{iter_{max}}$ has been exceed or when $\|\boldsymbol{r_{i+1}}\| \leq \boldsymbol{\varepsilon} \cdot \|\boldsymbol{r_i}\|$.

Each Newton-Raphson iteration adds $\boldsymbol{\alpha_i} \cdot \boldsymbol{d_i}$ to $\boldsymbol{z_i}$ and every addition from last steps have to fall below a tolerance $\boldsymbol{\epsilon}$ or it will just be setting as the boundary for each step addition. Newton-Raphson are terminated when the number of iterations exceeds $\boldsymbol{j_{max}}$

Keep updating new value of each attributes as long as an iteration is finished and no stopping criteria reached. Nonlinear conjugate gradient is restarted by setting $\boldsymbol{d_i} = \boldsymbol{r_i}$ whenever a search direction is computed not in descent direction $\boldsymbol{r_i^T} \cdot \boldsymbol{d_i} > \boldsymbol{0}$. It is also restarted once every $\boldsymbol{n}$ iterations, so as to improve convergence by setting a small $\boldsymbol{n}$. The complete flow for conjugate gradient algorithm is shown below.

$$r_i = -g_i \;\; ; \;\; h_i = \nabla g_i \;\; ; \;\; d_i = r_i \;\; ; \;\; \delta_{new} = r_i^T \cdot r_i \;\; ; \;\; \delta_0 = \delta_{new}$$

$$while\left(iter = \boldsymbol{iter_{max}} \;\; and \;\; \delta_{new} > \boldsymbol{\varepsilon^2} \cdot \delta_0\right)$$

$$do$$

$$\alpha_i = r_i^T \cdot d_i / d_i^T \cdot h_i \cdot d_i$$

$$z_{i+1} = z_i + \alpha_i \cdot d_i$$

$$if\left(|\alpha_i \cdot d_i| > \boldsymbol{\epsilon}\right)$$

$$\alpha_i \cdot d_i = \pm \boldsymbol{\epsilon}$$

$$j = j + 1$$

$$while\left(j < \boldsymbol{j_{max}}\right)$$

$$r_i = -g_i \;\; ; \;\; h_i = \nabla g_i \;\; ; \;\; \delta_{new} = r_i^T \cdot r_i \;\; ; \;\; \delta_{old} = \delta_{new}$$

$$\beta = \delta_{new} / \delta_{old} \;\; ; \;\; d_i = r_i + \beta \cdot d_i \;\; ; \;\; i = i + 1$$

$$if\left(i = \boldsymbol{n} \;\; or \;\; r_i^T \cdot d_i \leq 0\right)$$

$$d_i = r_i \;\; ; \;\; i = 0$$

$$iter = iter + 1$$

Fig. 12. Nonlinear Conjugate Gradient with Newton-Raphson and Fletcher-Reeves and bold font represent the customized parameter

6. EXPERIMENT SETUP

6.1. Benchmarks

We carried out several experiments results to prove the efficiency and quality of proposed algorithm. Our test cases are from ISPD2007 and ISPD2008 Global Routing Contest Benchmark. There are 16 benchmarks in total, which is from industrial ASIC designs and all of them consist of multi-metal layers. Note that 8 of the 16 benchmarks are from ISPD2007 3D version and the others are from ISPD2008 suite. Though ISPD2007 also have the version of 2D, we won't take too much concern about it. By observing these 16 benchmarks we are going to test, it's all multi-metal layer and 3D design; moreover, 3 of 16 benchmarks equip eight device layers, others just provide total six device layers as resource.

Inspired by ICCAD 2009 [9] as we mentioned in Section3.4, some benchmarks will include layer directive information for timing critical nets. This information shows these nets have tighter slack and have to be detoured to higher layer, where can make signal conduct faster so as to release the timing violation. Depending on different timing request, some may ask netbox goes to the highest layer; others only need not to be located at the lowest layer. We will name the netbox happen in former cases as critical nets, and later one as marginal critical nets respectively. Since ISPD2007 and ISPD2008 benchmarks don't have layer directive information included; besides, motivated by

ICCAD2009, we will randomly generate each of these 16 benchmarks with 20% of nets critical, 20% marginal critical and the left 60% with no layer directive preference given.

Though we randomly generate the new benchmarks with critical and marginal critical nets, we will still have only one version of benchmarks in specific test cases and we will call it modified benchmarks from here.

6.2. Comparisons

Since we generate modified benchmarks by our own, we will first make sure if all of them are executable by NTHU-route and by our layer planner. Table 1 shows the execution detail of NTHU-route and our layer planner on modified benchmarks. Here, we just show total 9 benchmarks that work under both algorithms. Some of the modified benchmarks cannot be routed completely by NTHU-route, since segmentation fault occurred. Others happened segmentation fault in our layer planner.

Table 1. Detailed execution results of proposed layer planner and NTHU-route

ISPD2008 BENCHMARK	g-cell_X dimension	g-cell_Y dimension	#Nets	#Marginal Critical Nets	#Critical Nets	OUR-layer planner Executable	OUR-layer planner TIME (iteration = 20)	NTHU-route Executable	NTHU-route TIME (iteration = 50+10)
adaptec1_modified	324	324	219794	43982	43802	Yes	13min	Yes	70min
adaptec2_modified	424	424	260159	52170	51747	Yes	11min	Yes	36min
adaptec5_modified	465	468	867441	173229	173332	Yes	45min	Yes	220min
bigblue1_modified	227	227	282974	56723	56327	Yes	8min	Yes	80min
newblue1_modified	399	399	331663	66526	65957	Yes	6min	Yes	30min
newblue2_modified	557	463	463213	92613	92264	Yes	12min	Yes	3min
newblue4_modified	455	458	636195	127174	126766	Yes	26min	Yes	72min
newblue5_modified	637	640	1257555	251295	251027	Yes	80min	Yes	190min
newblue6_modified	463	464	1286452	257081	256795	Yes	30min	Yes	210min

From Table 1, since our layer planner focus on different objectives and solution space are much smaller compared to NTHU-route, we can have these benchmarks properly done really fast in our algorithm. However, we are not showing comparison between our planner and NTHU-route, just to demonstrate the execution capability of these modified benchmarks.

Being a layer planner, we provide layer guidance for pre-routing optimization like buffer insertion. At the same time, as a pre-routing process also, we took routability into account in our design. Hence, we will address on following mentioned aspects to do comparison.

Firstly, we will try to demonstrate that when our layer planner algorithm are being inserted before NTHU-route, that means, when NTHU-route follow our netbox layer guidance first and then do NTHU-route, the routing attributes will be much better than purely executing NTHU-route.

Secondly, we will compare with some other layer planner and to see the performance of routing attributes. Nevertheless, there is no similar work focusing on the same issue till now, so we come up with running NTHU-route for a while and stop prematurely. The intermediate output of NTHU-route will be regarded as certain layer guidance, and we name it as NTHU-layer planner. By doing the same layer planner task as our layer planner and being fair for later comparison, our layer planner and NTHU-layer planner should run the modified benchmarks for the same time. After both of the layer planner finished, we will feed their layer guidance information back to

NTHU-route again and this time will exploit NTHU-route as router rather than stop at some point. After that, we compare both performances on routing attributes to see which layer planner provide better routing quality. The complete experimental comparison flow is shown in Fig. 13.

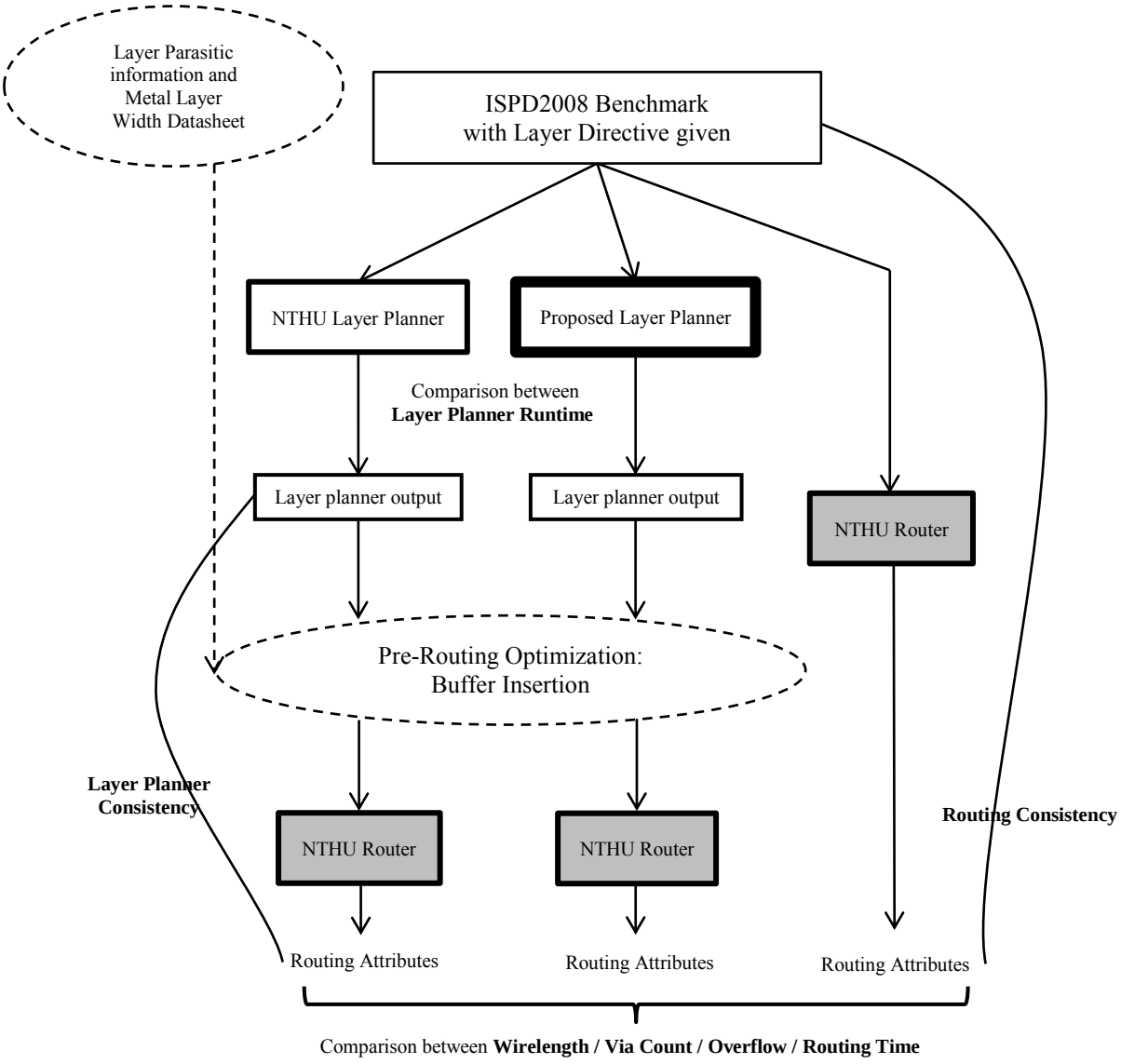


Fig. 13. Complete flow for experimental results comparisons

The dashed line in Fig. 13 is the part we didn't implement in this work just like we mentioned in Section2. But we will finally combine layer guidance and further buffer insertion algorithm together in the future to testify the buffer insertion ability after our layer guidance algorithm being applied in flow.

6.3. Layer Planner Output

The ISPD2008 modified benchmarks have layer directive information inside, which is adding by our own with randomly generated 20% marginal critical nets and 20% critical nets. The marginal critical nets means the layer directive information is given and saying that these nets should route in higher layer, and the critical nets represent the nets should route in the highest layer. In six metal device layer design, marginal critical information give the suggestion that we should route the nets between layer three and four, and critical one should route in layer five and six, which can be represented in file format as [3:4] and [5:6] respectively. Moreover, for eight metal device layer design, marginal critical and critical nets would be represented in file format as [5:6] and [7:8], which means former one is suggested to route between layer five and six but later one is layer seven and eight separately.

After feeding ISPD2008 modified benchmarks into layer planner, layer planner output will also provide layer guidance for each and every netboxes. This guidance will then overwrite the layer directive information and output another file with same format as modified benchmarks, only replacement on layer directive information to layer guidance information.

However, proposed planner being just a layer guider and the reason for not too much constraint for further NTHU-route, we won't give every netbox their layer guidance information to layer planner output. We only provide layer guidance for those who have to be put onto the highest two tier after layer planner results. That is, if a netbox in six metal layer design being assigned to layer three and four by layer planner, we will having [3:4] format to tell further NTHU-route the guidance for this nets. On the other hand, if a netbox in six metal layer design being assigned to layer one and two by layer planner, which is not in the highest two tier, we will not give any guidance for this nets in layer planner output, which is input for further NTHU-route. Fig. 14 below is to show an example about this.

	Layer directive information from modified benchmark		Assigned layer Derived from Layer planner algorithm	Layer guidance in layer planner output for further NTHU-route
netobx1	[3:4]	Layer Planner →	[1:2]	
netobx2			[3:4]	[3:4]
netobx3	[5:6]		[5:6]	[5:6]

Fig. 14. An example to show how layer guidance information replace the layer directive in modified benchmarks and how it displayed in layer planner output file for further NTHU-route

6.4. Consistency

After the layer planner output being routed by NTHU-route, we will have to compare the routing attributes with each different setup. Aside from routing attributes

along, in Fig. 13 the comparison flow, we may notice there are two new terminologies: Layer planner consistency and Routing consistency, which is also a comparison object in our experiment. We have to finish NTHU-route so that we can know the value for these two terms.

Layer planner consistency means after NTHU-route the ratio of wirelength which follow the layer guidance information derived from layer planner to total wirelength of that specific netboxes. On the other hand, Routing consistency represent after NTHU-route the ratio of wirelength which follow the layer directive information given from modified benchmarks to total wirelength of that specific netboxes. Fig. 15 is to show this concept.

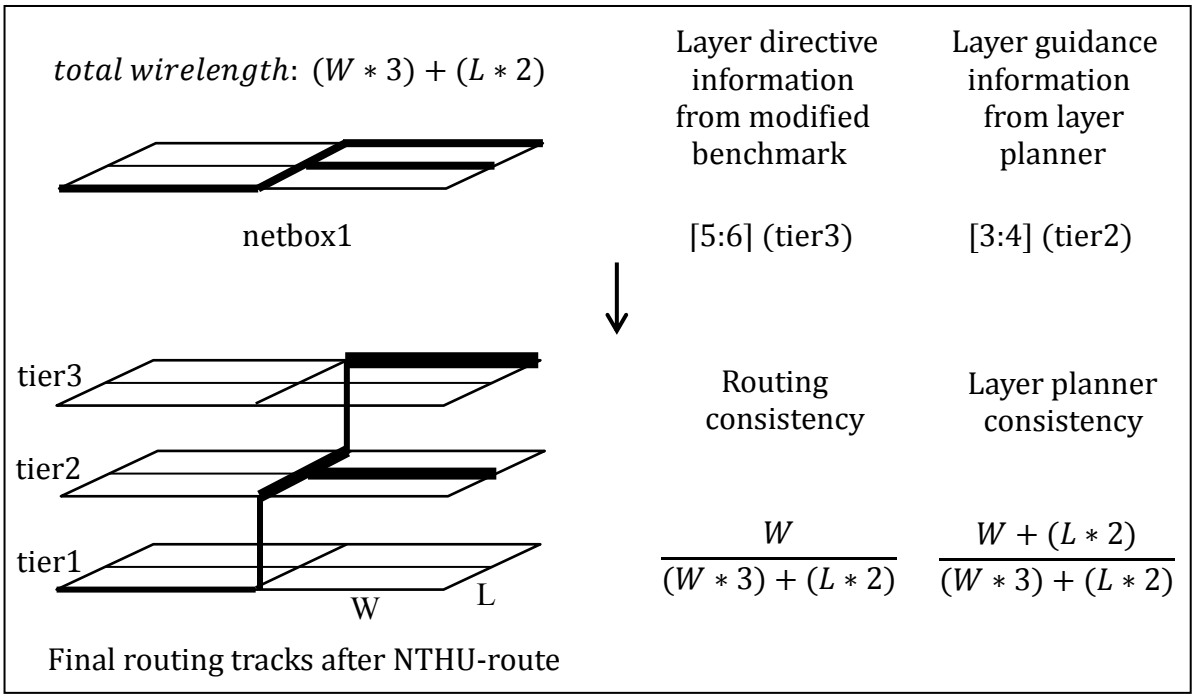


Fig. 15. An example to show how Routing consistency and Layer planner consistency calculated

7. EXPERIMENT RESULTS

The complete comparison flow could be consulted in Fig. 13. The comparison focus on how the layer planner affect the NTHU-route attributes; besides, see the different impact on routing attributes by our layer planner and NTHU-layer planner. However, in some cases, NTHU-route cannot generate valid output, we will use N/A to represent it in following tables, and use NULL represent that column won't have any value. Following tables are showing the comparison on ISPD2008 modified benchmarks.

Table 2. ISPD2008-adaptec2 modified benchmarks comparison

ISPD2008-adaptec2	NTHU-Layer Planner + NTHU-route	OUR Layer Planner + NTHU-route	NTHU-router
Layer Planner Runtime	8min	8min	NULL
Overflow after routing	3177	79	51
LP_Routing_Consistency	70%	44%	NULL
Routing_Consistency	43%	44%	43%
Wirelength + Via_count	3702243 + 1992554	3468009 + 1976324	3465751 + 1976215
NTHU-route Runtime	190min	41min	36min

Table 3. ISPD2008-newblue1 modified benchmarks comparison

ISPD2008-newblue1	NTHU-Layer Planner + NTHU-route	OUR Layer Planner + NTHU-route	NTHU-router
Layer Planner Runtime	6min	6min	NULL
Overflow after routing	2511	811	829
LP_Routing_Consistency	74%	46%	NULL
Routing_Consistency	43%	43%	43%
Wirelength + Via_count	2612930 + 2295545	2473652 + 2236884	2479568 + 2231928
NTHU-route Runtime	120min	22min	30min

Table 4. ISPD2008-newblue2 modified benchmarks comparison

ISPD2008-newblue2	NTHU-Layer Planner + NTHU-route	OUR Layer Planner + NTHU-route	NTHU-router
Layer Planner Runtime	1min	5min	NULL
Overflow after routing	100	0	0
LP_Routing_Consistency	68%	40%	NULL
Routing_Consistency	43%	41%	42%
Wirelength + Via_count	4970406 + 3004260	4830432 + 2992832	4790473 + 2988173
NTHU-route Runtime	90min	4min	3min

Table 5. ISPD2008-newblue4 modified benchmarks comparison

ISPD2008-newblue4	NTHU-Layer Planner + NTHU-route	OUR Layer Planner + NTHU-route	NTHU-router
Layer Planner Runtime	10min	10min	NULL
Overflow after routing	1267	627	772
LP_Routing_Consistency	76%	46%	NULL
Routing_Consistency	44%	44%	44%
Wirelength + Via_count	8706377 + 4809511	8657762 + 4796792	8724702 + 4811728
NTHU-route Runtime	73min	78min	72min

Table 6. ISPD2008-newblue5 modified benchmarks comparison

ISPD2008-newblue5	NTHU-Layer Planner + NTHU-route	OUR Layer Planner + NTHU-route	NTHU-router
Layer Planner Runtime	30min	30min	NULL
Overflow after routing	58639	332	610
LP_Routing_Consistency	68%	47%	NULL
Routing_Consistency	44%	44%	44%
Wirelength + Via_count	15792918 + 8683181	15578131 + 8815405	15813517 + 8891982
NTHU-route Runtime	720min	130min	190min

Above five benchmarks start from Table 2 to Table 6 are the most complete few, since both layer planner work fine and no segmentation fault occurred. Besides, NTHU-route can also execute both layer planner's output and process them without any error.

By observing the results, with our layer planner process first and then through layer guidance result to NTHU-route, the routing attributes become much better. For example, in modified benchmark ISPD2008-newblue5, with benchmark direct disposed by NTHU-route, it will take 190 minutes to finish routing. On the other hand, if we assert modified benchmark with our layer planner first then feed output with layer guidance to NTHU-route, it now can have better routing quality since taking only 130 minutes and better wirelength, via count and overflow counted.

Furthermore, by comparing each layer planner quality, we can easily find out that to be fair, if we fix both layer planner runtime as the same, our layer planner results with layer guidance information can make NTHU-route generate much better routing quality in terms of all the routing attributes.

So we can say, our layer planner can improve NTHU-route ability by providing it with promising layer guidance for modified benchmarks. Besides, under the circumstance being fair in both planner runtime, our layer planner provides effective layer guidance for further NTHU-route compared to NTHU-layer planner suggested.

Table 7. ISPD2008-adaptec1 modified benchmarks comparison

ISPD2008-adaptec1	NTHU-Layer Planner + NTHU-route	OUR Layer Planner + NTHU-route	NTHU-router
Layer Planner Runtime	10min	10min	NULL
Overflow after routing	N/A	1	3
LP_Routing_Consistency	N/A	44%	NULL
Routing_Consistency	N/A	47%	46%
Wirelength + Via_count	N/A	39173113 + 1940287	3953906 + 1958280
NTHU-route Runtime	N/A	43min	70min

Table 8. ISPD2008-adaptec5 modified benchmarks comparison

ISPD2008-adaptec5	NTHU-Layer Planner + NTHU-route	OUR Layer Planner + NTHU-route	NTHU-router
Layer Planner Runtime	25min	25min	NULL
Overflow after routing	N/A	3	386
LP_Routing_Consistency	N/A	44%	NULL
Routing_Consistency	N/A	45%	45%
Wirelength + Via_count	N/A	11037720 + 5547420	11154398 + 5542447
NTHU-route Runtime	N/A	150min	220min

Table 9. ISPD2008-bigblue1 modified benchmarks comparison

ISPD2008-bigblue1	NTHU-Layer Planner + NTHU-route	OUR Layer Planner + NTHU-route	NTHU-router
Layer Planner Runtime	10min	10min	NULL
Overflow after routing	N/A	0	214
LP_Routing_Consistency	N/A	46%	NULL
Routing_Consistency	N/A	46%	45%
Wirelength + Via_count	N/A	3969670 + 2050326	4157406 + 2138076
NTHU-route Runtime	N/A	30min	80min

Table 10. ISPD2008-newblue6 modified benchmarks comparison

ISPD2008-newblue6	NTHU-Layer Planner + NTHU-route	OUR Layer Planner + NTHU-route	NTHU-router
Layer Planner Runtime	30min	30min	NULL
Overflow after routing	N/A	1	666
LP_Routing_Consistency	N/A	45%	NULL
Routing_Consistency	N/A	45%	44%
Wirelength + Via_count	N/A	11319829 + 7998592	11636380 + 8085488
NTHU-route Runtime	N/A	110min	210min

Observing from Table 7 to Table 10, they are also quite complete in experimental statistic. Both layer planner works fine without error occurred; however, when we try to feed NTHU-layer planner output with layer guidance to further NTHU-route, the segmentation fault comes out. This kind of segmentation fault is hard to detect, since NTHU-route are not supposed to be used as a layer planner when it was invented; besides, in great work like NTHU-route, it is really hard to debug. As a result, we cannot have our comparison between layer planner. But concluded from Table 2 to Table 6, we can definitely know that the layer guidance information from our layer planner will out win NTHU-layer planner.

With above four benchmarks results, we can still conclude again that modified benchmarks with our layer planner asserted will generate more quality routing results in terms of wirelength, via count, overflow and runtime compared to direct feeding modified benchmark to NTHU-route. So our layer planner still output promising and quality layer guidance for NTHU-route by above observation.

8. CONCLUSION

In typical physical design flows, many optimizations are performed between the placement and routing, such as gate sizing and buffer insertion. Our objective is to develop a pre-routing analytical layer planner, which can efficiently generate guidance for further buffer insertion. As a step before routing, we also need to consider routability so that results can lead to routable circuits.

We also noticed nowadays trend of following layer directive information to relief timing violation by considering BETA factor. Besides, the number of metal layers keeps increase and metal size/parasitic among different layers becomes increasingly non-uniform. We tackle this by taking RUDY factor into account.

At the end, we assert nonlinear conjugate gradient method to do optimization and the solution give each netbox an integer tier to place. By inserting our algorithm before NTHU-route, the experimental results show that our approach do provide easier and efficient routing, in terms of smaller wirelength, total overflow and less runtime using, compared with only asserting NTHU-route on same benchmark. Our approach has achieved significant success in providing quality layer planning guidance for buffer insertion; while, at the same time, generate efficient guidance for further routing.

REFERENCES

[1] J. Cong, G. Luo, "A multilevel analytical placement for 3D ICs," *in Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, January 19-22, 2009, Yokohama, Japan

[2] P. Spindler, F.M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," *in Proceedings of the conference on Design Automation and Test in Europe*, April 16-20, 2007, Nice, France

[3] Z.W. Jiang, B.Y. Su, Y.W. Chang, "Routability-driven analytical placement by net overlapping removal for large-scale mixed-size designs," *in Proceedings of the 45th annual conference on Design Automation*, June 08-13, 2008, Anaheim, California

[4] A.B. Kahng, Q. Wang, "Implementation and extensibility of an analytic placer," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24. no. 5, pp. 734-747, 2005

[5] Y.J. Chang, Y.T. Lee, T.C. Wang, "NTHU-Route 2.0: a fast and stable global router," *in Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, November 10-13, 2008, San Jose, California

[6] Y.J. Chang, T.H. Lee, T.C. Wang, "GLADE: a modern global router considering layer directives," *in Proceedings of the International Conference on Computer-Aided Design*, November 07-11, 2010, San Jose, California

[7] T.H. Lee, Y.J. Chang, T.C. Wang, "An enhanced global router with consideration of general layer directives," *in Proceedings of the International Symposium on Physical Design*, March 27-30, 2011, Santa Barbara, California

[8] M. Pan, C. Chu, "FastRoute 2.0: A High-quality and Efficient Global Router," *in Proceedings of the 2007 Asia and South Pacific Design Automation Conference*, January 23-26, 2007, Yokohama, Japan

[9] M.D. Moffitt, "Global routing revisited," *in Proceedings of the 2009 International Conference on Computer-Aided Design*, November 02-05, 2009, San Jose, California

[10] T.H. Lee, T.C. Wang, "Congestion-constrained Layer Assignment for Via Minimization in Global Routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 9, pp. 1643-1656, 2008

[11] G.J. Nam, J. Cong, "Modern Circuit Placement: Best Practices and Results," *Springer Publishing Company*, New York, 2007

[12] T.C. Chen, Z.W. Jiang, T.C. Hsu, H.C. Chen, Y.W. Chang, "A high-quality mixed-size analytical placer considering preplaced blocks and density constraints," *in Proceedings of the International Conference on Computer-Aided Design,* November 05-09, 2006, San Jose, California

[13] J. Nocedal, S.J. Wright, "Numerical Optimization 2nd ed," *Springer Publishing Company*, New York, 2006

[14] C. Li, M. Xie, C. Koh, J. Cong, P. Madden, "Routability-driven placement and white space allocation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no.5, pp.167-172, May 2008

[15] G.J. Nam, "ISPD 2006 Placement Contest: Benchmark Suite and Results," *in Proceedings of the International Symposium on Physical Design*, April 09-12, 2006, San Jose, California

[16] A.B. Kahng, S. Reda, Q. Wang, "Architecture and details of a high quality, large-scale analytical placer," *in Proceedings of the International Conference on Computer-Aided Design*, November 06-10, 2005, San Jose, California

[17] A.B. Kahng, X. Xu, "Accurate pseudo-constructive wirelength and congestion estimation," *in Proceedings of the 2003 international workshop on System-level interconnect prediction*, April 05-06, 2003, Monterey, California

[18] J.R. Shewchuk, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain," 1994, Pittsburgh, Pennsylvania

[19] M.S. Bazaraa, H.D. Sherali, C.M. Shetty, "Nonlinear Programming: Theory and Algorithms 3rd ed," *John Wiley & Sons Inc.*, New Jersey, 2006

[20] K.G. Murty, "Linear Complementarity, Linear and Nonlinear Programming," *Helderman Verlag*, Berlin, 1988

VITA

Chi-Yu Chang received his Bachelor of Engineering degree in electrical engineering from National Chung Cheng University, Chiayi, Taiwan in 2009. After one year of military service, in August 2010, he joined the Texas A&M University and graduated with Master of Science degree in computer engineering in August 2012. His research interests include digital integrated circuit design, physical design and VLSI computer aided design. He can be reached at the Department of Electrical and Computer Engineering, 238 Zachry Engineering Center, Texas A&M University, College Station, TX 77843-3128. His email is *changalvin@tamu.edu*.