A LAYER CENTRIC VLSI PHYSICAL DESIGN METHODOLOGY CONSIDERING NON-UNIFORM METAL STACKS

A Thesis

by

SITONG ZHAI

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

MASTER OF SCIENCE

Chair of Committee,	Jiang Hu
Committee Members,	Weiping Shi
	Duncan M. (Hank) Walker
Head of Department,	Miroslav M. Begovic

August 2015

Major Subject: Computer Engineering

ABSTRACT

VLSI technology scaling has caused interconnect delay to increasingly dominate the overall chip performance. Optimization techniques such as buffer insertion, wire sizing and layer assignment play critical roles in successful timing closure for chip designs. For several VLSI technology generations, designers have confronted the challenges associated with increasing wire delays. One industrial solution is to add layers of thicker metal to the wiring stacks. However, the existing physical synthesis tools are not effective enough to handle these new thick metal layers. Thus, it is necessary to design a new flow to provide better communication among layer planning, buffering, routing and different optimization engines. In this thesis, our work proposes a new design flow, Layer Centric Design Flow, to perform congestion mitigation and timing optimization with layer directives. Our design flow balances buffer and routing resources so that the design benefits from the availability of thick metal layers and reduces buffer usage while maintaining routability as well as performance.

DEDICATION

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.

Firstly, I would like to give 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. Duncan M. (Hank) Walker for being my committee members, and for their suggestions on this research.

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.

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

Page

1. INTRODUCTION

1.1 Physical Design in VLSI and Motivation

Physical design directively impacts circuit performance, area, reliability, power and manufacturing yield. Due to its high complexity, physical design is usually partitioned into several key steps: floor planning, placement, buffering, global routing and detailed routing. VLSI technology scaling has caused interconnect delay to increasingly dominate the overall chip performance. Physical synthesis is a core component of modern VLSI design methodologies for design closure. However, existing methodologies are inefficient at handling the non-uniform metal layers.

Interconnect delay has become a major influence in the overall propagation delay in the modern Deep Sub-Micron technology (DSM) designs as process technology advances. The focus of design methodology is gradually shifting from logic optimization to interconnect optimization. In conventional design flows, many interconnect optimizations are performed between the placement and routing. Especially, buffer insertion is a very efficient way to improve circuit performance. The mainstream of most interconnect optimization techniques remains to be buffer insertion, because it is easy to apply and can effectively reduce the interconnect delay. Thus this technique is widely used in modern VLSI designs to achieve better performance. As new nanometer process technology with multiple routing layers becomes available [1] and metal parasitic among different layers becomes increasingly non-uniform, layer and its corresponding parasitic

information are necessary in these optimization techniques. Without proper estimation of layer and corresponding parasitic information, tools may insert huge amount of buffers during the interconnect optimization, and cause more power consumption and greater area of chip. Although assigning to thick metal results in efficient timing, routability of the design may be compromised.

1.2 Non-uniform Metal Layers and Routing Track

There are four 1x metal layers, three 2x metal layers and two 4x metal layers at IBM 65nm technology [8] as shown in Fig. 1.

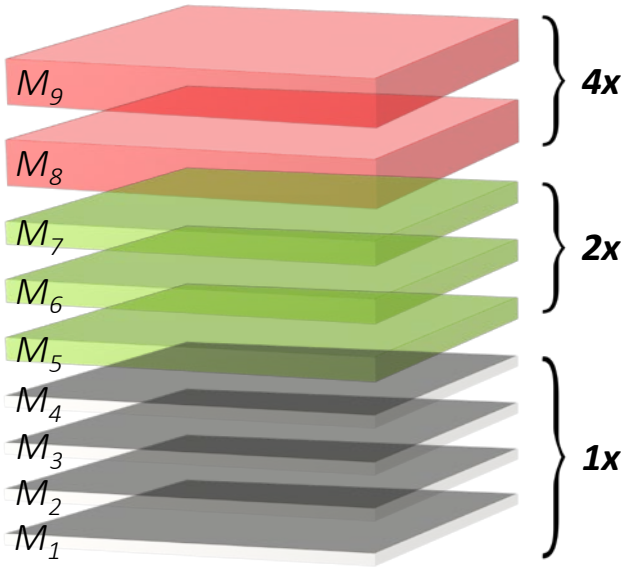

Fig. 1. A pictorial view for metal layers of the 65nm technology [8]

Fig. 2 shows a cross section of a routing tile with 7 routing tracks in 65nm node, of which four belong to a single 1x thickness layer, two belong to a double 2x thickness layer, and one belongs to the top 4x thickness layer. When assigning a wire in this tile to the thick metal for the best performance, there is one thickest metal available to choose from.

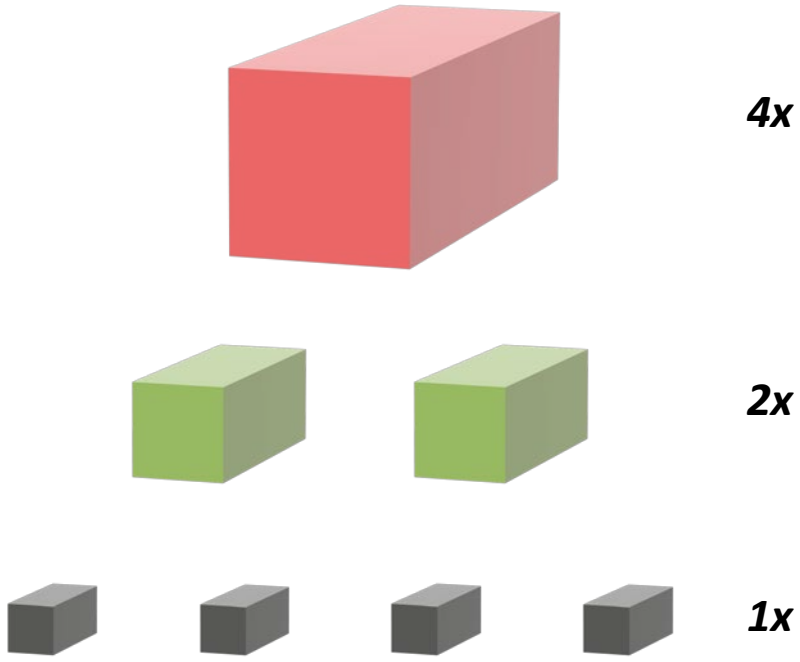

Fig. 2. Routing track for cu65nm

Layer assignment, if properly performed, can reduce both interconnect delay and the number of buffers needed. As shown in Fig. 2, the wire on upper layer is both wider and thicker. Assuming that wire width changes at the same ratio as the wire thickness,

resistance per unit length on 2x [4x] thickness layer is roughly 1/4 [1/16] of resistance per unit length on 1x thickness layer; capacitance per unit length on 2x [4x] thickness layer is roughly twice [four times] of capacitance per unit length on 1x thickness layer, signal travels 2x faster for 2x thickness layer and 4x faster for 4x thickness layer. In reality, as a result of input slew, non-zero buffer intrinsic delay and wire capacitance change, the above number could be off but the benefit of layer assignment is still tremendous.

On the 2x [4x] layer, signals can roughly go 1.7x [2.5x] faster with 2x [4.4x] reduction in buffer resources [8]. Therefore, assigning timing critical nets to thick layers can reduce area/power and improve timing closure by reducing delays and the buffer count.

As investigated in [8], the slack for a two-pin net on a 4x layers is improved from -10ps to 10ps compared with the corresponding route on the 1x layer, and the number of buffers is reduced from 7 to 1, which is shown in Fig. 3.

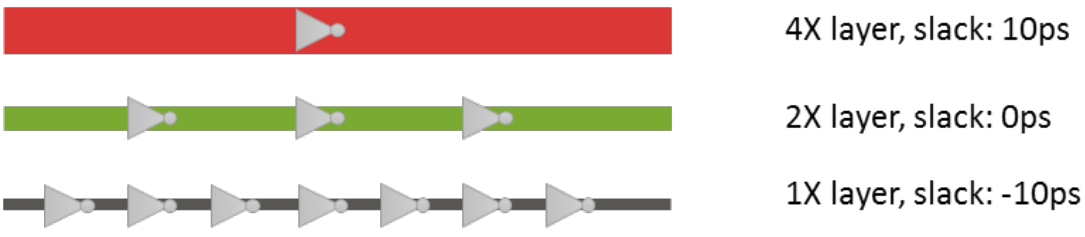

Fig. 3. Assigning the same net to thicker layers improves timing and buffering [8]

1.3 Conventional VLSI Physical Design Flow

In conventional VLSI physical design flow shown in Fig. 4, buffer insertion is performed before routing and therefore has no knowledge of wire layer information. Average metal layers information such as average resistance and average capacitance are used for buffer insertion stage, which results in over-buffering. About 30% reduction on buffer area averaged over several industrial circuits can be achieved by using thick layers wisely. On the other hand, the thicker layer resources are limited, the design may not be routable if too many nets are assigned to thicker layers. Besides reducing buffers, we need to carefully use the limited thicker layer resources.

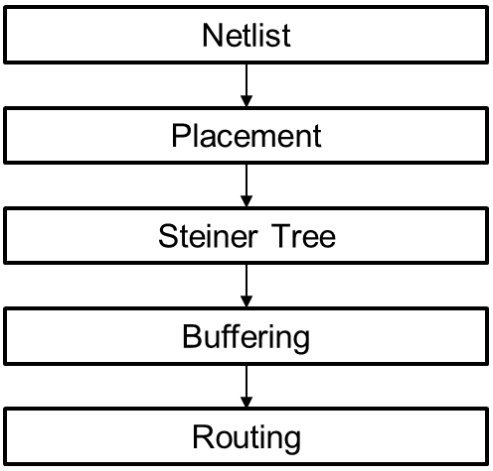

Fig. 4. Conventional VLSI physical design flow

1.4 Our Contributions

We propose a new design flow, Layer Centric Design Flow, to reduce the usage of buffers and thick layer while efficiently utilize the interconnect timing budget. Different from the conventional design flow in Fig. 4, we address the early planning for layer assignment before the buffering stage. Then we take the early layer assignment results for simultaneous buffering and layer assignment stage and use buffering and layer assignment results as guidance for global routing. Compared with the previous design flow, our approach achieves about 40% reduction in buffer resource and 0.2% reduction in wire resource.

1.5 Outline

The remainder of this thesis is organized as follows. Section 2 is to introduce the layer directives and notations. Section 3 introduces the previous work. Section 4 presents our Layer Centric Design Flow and gives illustration on each stage. Section 5 shows our experimental results. Section 6 concludes this thesis.

2. PRELIMINARY

As shown in Fig. 1, in 65nm technology, there are three different planes (a plane refers to the layers with same parasitic): $[M_1, M_2, M_3, M_4]$ are in the lower layer which have the thinnest metal; $[M_5, M_6, M_7]$ are in the middle layer which are 2x thicker than 1x; and $[M_8, M_9]$ are in the upper layer which have the thickest metal. Here M_i denotes the i^{th} metal layer. The layer directive means a soft constraint on a wire which the valid layers can be placed on. For global routing, our layer planning and physical synthesis flow can provide layer directives. For example, $[M_5, M_7]$ for a wire means this wire is preferred to be routed only in metal layer M_5, M_6, M_7.

In this thesis, we generate a layer directive for each net from layer planning, then we generate the specific layer directive for each wire in this net from simultaneous buffering and layer assignment based on the layer planning result. For example, we generate a layer directive $[\boldsymbol{M_1}, \boldsymbol{M_4}]$ for a net from layer planning step, and then we get a directive $[\boldsymbol{M_5}, \boldsymbol{M_7}]$ for a wire in this net after simultaneous buffering and layer assignment step. This means our layer planning and physical synthesis flow provides this wire a 2x layer directive for global routing. Furthermore, the wire can also be routed on upper layers because the wire has less delay on thicker metal layers.

3. PREVIOUS WORK

Conventional routers perform layer assignment purely for routing congestion minimization and many works focus on how to perform layer assignment with via minimization [2, 6]. In these works, the timing benefit of thick layers is not leveraged. Subsequent work on timing-driven layer assignment [1, 6, 7] used timing information to drive layer assignment. As it might be necessary to consider layer assignment during routing, the timing gain in these works is limited since routing is performed after all optimizations are completed, or at least after a majority of buffers are placed.

As illustrated in section 1.2, assigning nets to thicker layers can improve timing and buffering. The difference in wire delays among different layers provides another dimension to timing optimization, besides sizing and buffering. Assigning timing-critical nets to thicker layers can improve timing closure and reduce the number of buffers.

An algorithm, CATALYST: Congestion And Timing Aware Layer Assignment, is proposed in [4]. CATALYST is to perform layer assignment to maximize the timing benefits with congestion control at early stages and is inserted into the conventional physical synthesis flow before global buffering. This algorithm tries to assign a large number of nets to thick layers with the goal to minimize the buffer usage and control the congestion when performing layer assignment. [4]

CATALYST algorithm is an iterative process. CATALYST algorithm constructs the minimal Steiner tree and apply 2D-routing for nets. Then, the nets will be assigned to higher layers until the scores of these net down to criteria. The score function is

described in (1). $s(n_i)$ denotes the worst slack of all the sinks of net n_i, T_C denotes the clock period used for normalization.

$$\boldsymbol{w(n_i) = exp\left(-\frac{s(n_i)}{T_C}\right)} \quad (\mathbf{1})$$

CATALYST algorithm generates initial layer directive assignment solution to meet the timing constraints. It proposes a simple timing-driven directive assignment heuristic by promoting the timing-critical nets to higher directives one by one with incremental timing updates. Then, it examines the initial directives and relax the initial directive of the net which causes congestion. 2D-routing will be rerun for each group of nets and congestion will be checked again after directive assignment adjusted. It reruns timing analysis based on the new directive assignments in each iteration. However, in the Layer Centric Flow, the simultaneous buffering and layer assignment step applies the 3D-routing, furthermore, we can get the layer assignment of each wire segment.

4. LAYER CENTRIC DESIGN FLOW

4.1 Overview of Layer Centric Design Flow

Layer Centric Design Flow (Fig. 5) is to perform congestion mitigation and timing optimization with layer directives. Our flow balances buffer and routing resources so that the design benefits from the availability of thick metal layers and reduces buffer usage while maintaining routability as well as performance. Different from the conventional design flow, our flow addresses the early planning for layer assignment prior to the buffering stage and takes the layer guidance information from simultaneous buffering and layer assignment for the following global routing.

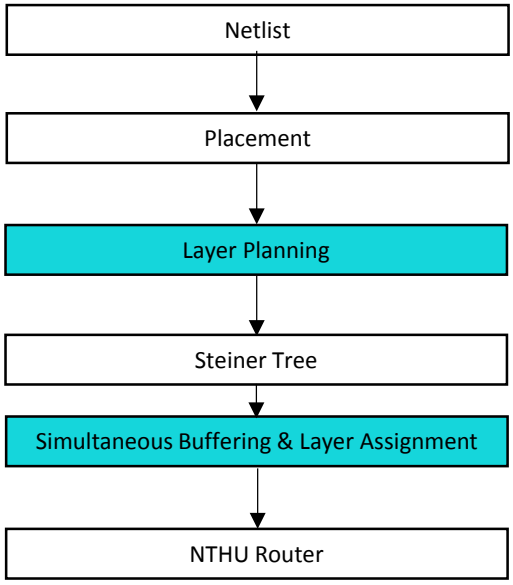

Fig. 5. Layer centric design flow

There are four main parts in layer centric design flow: Layer Planning, Steiner Tree Construction, Simultaneous Buffer Insertion and Layer Assignment, and the Global Routing [5].

Steps before routing like buffer insertion have to be combined with estimation of routability and parasitic so that the routing is not too difficult. The objective of layer planning is to provide layer-aware wire congestion estimation to buffer insertion. This planning provides each net with layer range where it should lie in; while, at the same time, keeps the routability issue and timing information into consideration. We use an analytical layer planning [9] for our flow.

In our work, we adopt FLUTE algorithm [3] to generate the Rectilinear Minimal Steiner Tree (RMST). The FLUTE algorithm uses a pre-computed lookup table to construct RSMT very quickly and can obtain the optimal solutions for low-degree nets. The nets with degree higher than 9 are broken into several sub-nets with degree ranging from 2 to 9 to avoid huge CPU time and memory consumption.

In the simultaneous buffering and layer assignment step, we adopt the Lagrangian relaxation and dynamic programming method to minimize the usage of buffer and wire resources while meeting the timing constraints.

NTHU-router [2] considers the layer guidance information, we use NTHU-router for the global routing process, which is based on iterative rip-up and reroute with a history based cost function to help distribute overflow. It has a congestion region identification method to specify the order for nets to be rip-up and reroute.

4.2 Layer Planning

The objective of layer planning is to find a pre-routing wire congestion estimation, which is useful for buffer insertion. This planner assigns each net to a layer range. 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.

What we want to fulfill is giving each netbox a layer range to lie in. In the layer planning step, we minimize the netbox overlap area by distributing them into different layer. The objective function is formed in (2):

$$\boldsymbol{Objective\ function}$$

$$= \sum \Big(\boldsymbol{netbox_{overlap}}(\boldsymbol{z_i}) \times \boldsymbol{Relaxation}(\boldsymbol{z_i}) \times \boldsymbol{RUDY}(\boldsymbol{z_i}) \times \boldsymbol{\beta}(\boldsymbol{z_i}) \Big) \big|_{\boldsymbol{z_i} \in [\mathbf{1}, \mathbf{2}, \ldots, \boldsymbol{M}]} \quad (\mathbf{2})$$

Besides minimizing overlap, consideration on timing critical issue is implemented by adding a β factor in the objective function. The β of a critical net is set to greater value if it is assigned to lower layers. Another factor RUDY stands for Rectangular Uniform wire DensitY and is defined as the ratio of the wire area to the nextbox area. RUDY facilitates netbox to go to lower layer in normal cases unless it is necessary to route in higher layer for reducing overlap or improving timing. The relaxation $(\boldsymbol{z_i})$ is to relax the discrete layer variable to continuous, so that the problem

can be solved by nonlinear programming solvers. The relaxed continuous results after optimization have to be mapped back to the most closed discrete value at the end.

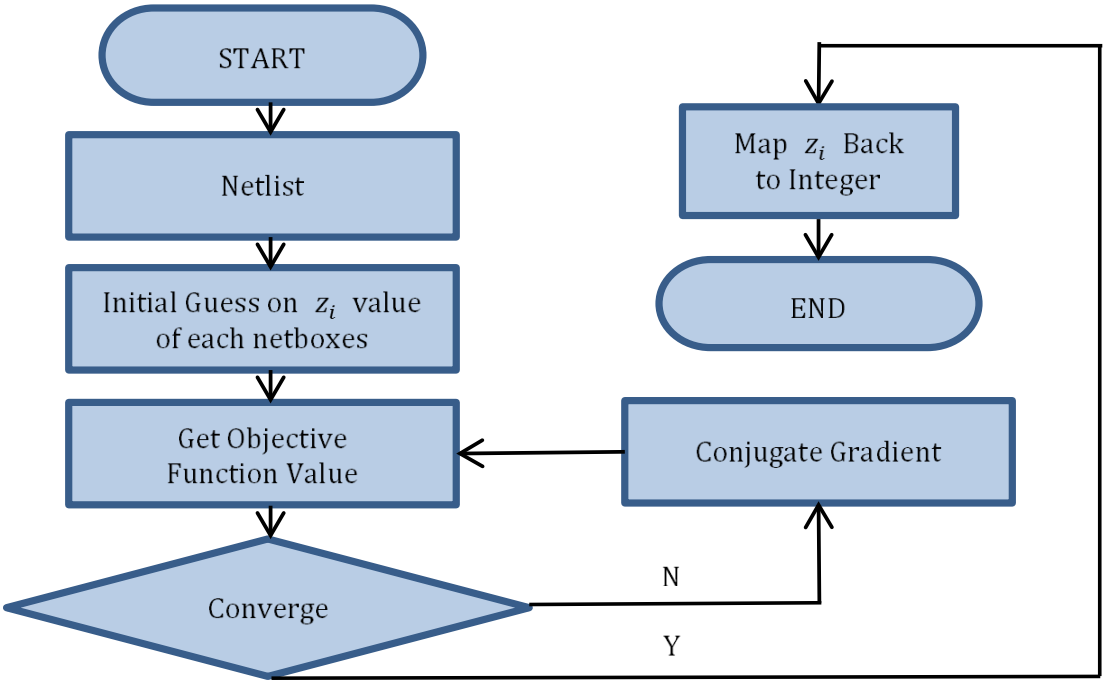

Fig. 6. Layer planner algorithm flow

Conjugate gradient method is used to minimize the objective function. Generally, the conjugate gradient method finds the optimum value by executing a series of line search. The result of one line search is used as the start point for the next line search till convergence. The overall layer planner flow is shown in Fig. 6.

4.3. Simultaneous Buffering and Layer Assignment

We optimize buffer resource considering non-uniform metal stacks in the simultaneous buffering and layer assignment step [10]. Fig. 7 shows the structure of a multi-layer interconnect tree $\mathbf{T} = (\mathbf{V}, \mathbf{E})$ where V is a set of nodes and E is a set of edges. Also, each edge e consists of several segments. Each segment s can be routed on any specific layer among multiple layer options. In this thesis, we adopt the Lagrangian relaxation method to minimize the usage of buffer and metal resources while to meet the timing constraints. The formulation of the objective function and Lagrangian relaxation method detail will be illustrated in section 4.3.2.

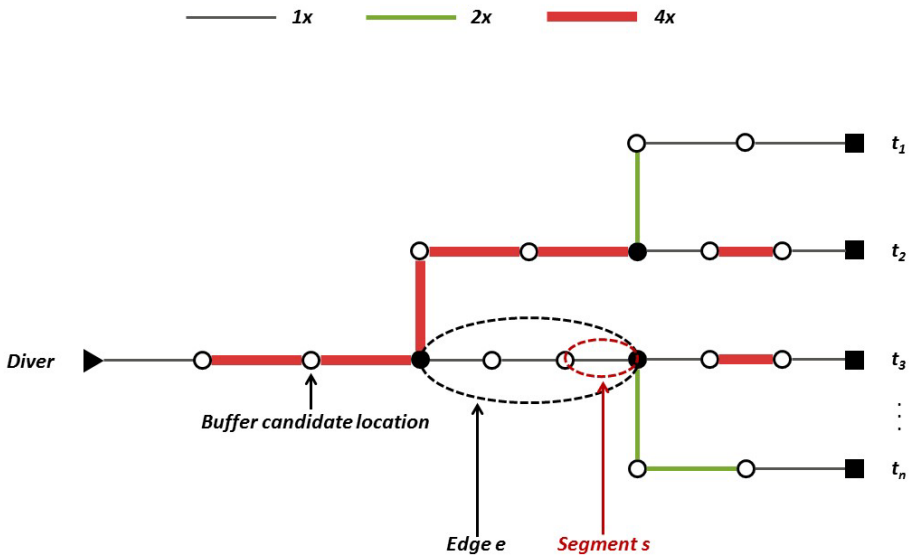

Fig. 7. Multi-layer interconnect tree

After buffer insertion and layer assignment stage, we get the specific layer directives for segments in these nets, and we separate these nets into some sub-nets shown in Fig. 8, and the segments in each sub-net have same layer directives. That is to say, once the segments connect with each other in the same layer, they will form one sub-net. As illustrated in Fig. 8, the circuit can be separated into 12 sub-nets for the following global routing. Thus, this net will be replaced by 12 nets with their own net layer directives in the benchmark, and the nets will be separated again into two-pin nets used for the following step global routing.

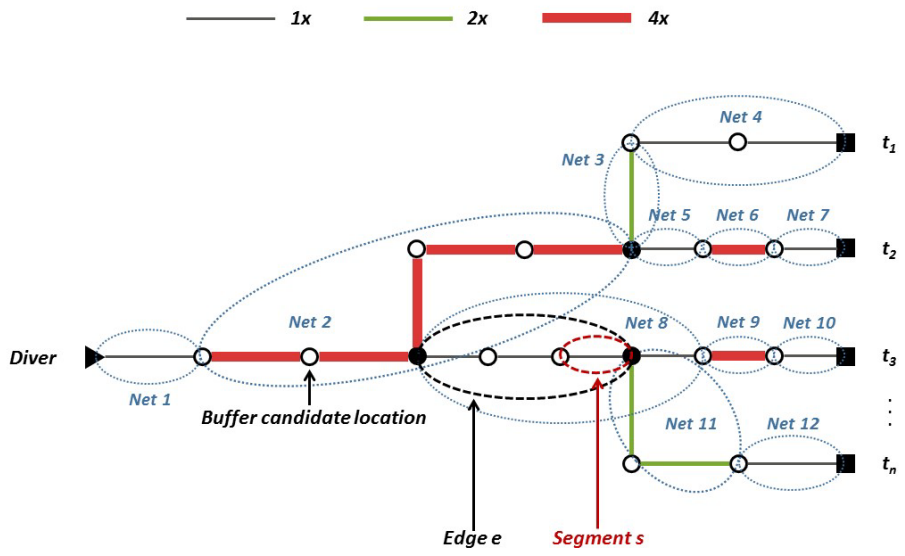

Fig. 8. Example of separated multi-layer interconnect tree

4.3.1. Coupling Capacitance

Interconnect parasitic parameters in integrated circuits have significant impact on circuit speed. In this thesis, we calculate wire capacitance with considering both ground capacitance and coupling capacitance. Same ground capacitance per unit length is used for the metal wire in the same layers. However the coupling capacitance is more difficult to calculate. In this case, we estimate it based on the wire density of grids as illustrated in Fig. 9. The layer guidance information of each net has been generated from layer planning step. Then, we can get the wire density of each grid after rectilinear minimal Steiner trees constructed on their corresponding layers with the layer directives obtained from layer planning.

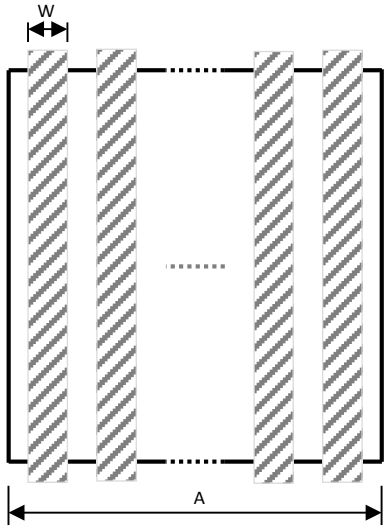

Fig. 9. Illustration for estimating coupling capacitance

Thus, we estimate the coupling capacitance as:

$$\mathbf{C_c} = \frac{\propto}{\frac{\mathbf{A} - \mathbf{K} \cdot \mathbf{W}}{\mathbf{K}}} = \frac{\propto \cdot \mathbf{K}}{\mathbf{A} - \mathbf{K} \cdot \mathbf{W}} \qquad (\mathbf{3})$$

Where A is the width of each grid, W is the width of wire in this layer, K is the number of wire on each grid boundary. $\propto$ is a constant of each grid.

4.3.2. Minimization of Buffer and Wire Resources

In In this thesis, we adopt the Lagrangian relaxation method to minimize the usage of buffer and metal resources while to meet the timing constraints [10]. Given a set of M routing layers $\boldsymbol{z_1, z_2, \ldots\ z_M}$ where $\boldsymbol{z_1}$ is the lowest layer and $\boldsymbol{z_M}$ is the highest layer. In Fig. 10, $\boldsymbol{l_s}$ is the length of segment s on layer $\boldsymbol{z_i}$, $\boldsymbol{r_s}$ is the resistance and $\boldsymbol{c_s}$ is the capacitance. The required arrival time are and downstream capacitance are given. Besides, we set the buffer candidate location number of each edge. Then, we perform the simultaneous buffer insertion and layer assignment on each edge to minimize the total buffer capacitance and metal capacitance.

In Fig. 10, $\boldsymbol{u_s}$ is the beginning point of edge $\boldsymbol{e}$ and $\boldsymbol{v_e}$ is the end point. Let $\boldsymbol{X_e}$ be the total buffer capacitance inserted on edge $\boldsymbol{e}$, and let $\boldsymbol{Y_e}$ be the total wire capacitance used for edge $\boldsymbol{e}$, we consider both ground capacitance and coupling capacitance in this

thesis. $\boldsymbol{d_e}$ denotes the signal delay along edge $\boldsymbol{e}$, $\boldsymbol{a_k}$ as the required arrival time at node $\mathbf{k} \in \mathbf{V}$, and $\boldsymbol{q_i}$ as the request arrival time for sink $\boldsymbol{t_i}$.

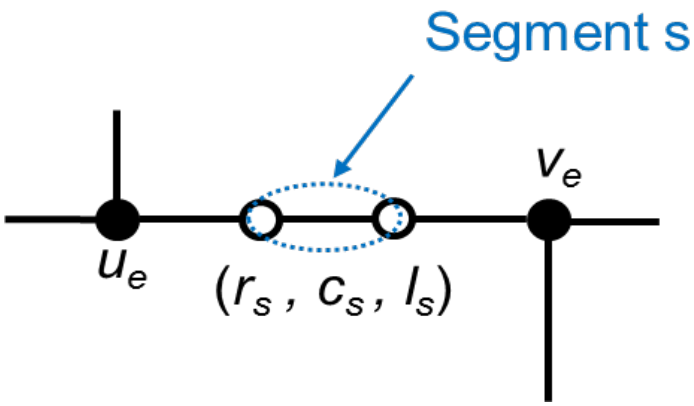

Fig. 10. A segment in an edge of a net [10]

Then, the problem can be formulated as (4) by combined buffer and wire capacitance :

$$\boldsymbol{Min} \sum_{\boldsymbol{e \in E}} \boldsymbol{\alpha X_e + \beta Y_e} \qquad (\mathbf{4})$$

Subject to

$$\boldsymbol{a_{src}} \geq \mathbf{0}$$

$$\boldsymbol{a_{u_e} + d_e \leq a_{v_e}}$$

$$\boldsymbol{\forall v_e \in sinks, a_{v_e} = q_i}$$

Where α and β are constant weight factors.

Then Lagrangian relaxation function can be written as (5):

$$\boldsymbol{L(\vec{X}, \vec{Y}, \vec{a}, \vec{\lambda}) = \sum_{e \epsilon E} \alpha X_e + \beta Y_e + \sum_{e \epsilon E} \lambda_e (a_{u_e} + d_e - a_{v_e}) - \lambda_{src} a_{src}} \qquad (5)$$

Where $\vec{X}$ is the vector of total buffer capacitance of X_e , $\vec{Y}$ is the vector of total metal capacitance of $Y_e, \forall e \in E$, $\vec{a}$ are the arrival times of all nodes, $\vec{\lambda}$ is the Lagrangian multiplier vector and λ_{src} is the Lagrangian multiplier for constraint $\lambda_{src} \geq 0$, λ_e is the Lagrangian multiplier of edge e for constrain $a_{u_e} + d_e \leq a_{v_e}$, and $a_{v_e} = q_i$ for $v_e = t_i$.

Then we solve the problem of simultaneous buffer insertion and layer assignment based on the method [10]. First, the Lagrangian multiplier is initialized. Under the given multiplier values during each iteration, the Lagrangian relaxation function is minimized using dynamic programming algorithm. We adopt the ellipsoid method is used to update the Lagrangian multipliers $\vec{\lambda}$. After buffer insertion and layer assignment, layer directives for wire segments in each net are obtained. Then each net was broken into several sub-nets which share same layer directive.

5. EXPERIMENT RESULTS

We implement all the algorithms in C++ language and execute it on the Linux operating system with a Dell PowerEdge R815 with 4 AMD Operon 6174 Processors (48 2.2GHz cores) and 256GB of Memory running CentOS5.7 x86_64.

We carried out several experiment results to prove the efficiency and quality of proposed algorithm. Our test cases are from ISPD2008 Global Routing Contest Benchmark. There are 8 benchmarks in total, which are from industrial ASIC designs and all of them have multi-metal layers. Table 1 shows the metal usage distribution and the worst negative slack of our layer centric flow. Tables 2-11 show the comparison on conventional and Layer Centric Flow results.

Table 1. Metal usage distribution and worst negative slack of Layer Centric Flow

ISPD2008 BENCHMARK	#Nets	Grid	Worst slack(ps)	Layer Distribution		
				1x	2x	4x
adaptec2_modified	260159	424×424	-132.90	69.13%	22.46%	8.41%
adaptec3_modified	466295	774 ×779	-177.57	69.33%	25.25%	5.42%
adaptec4_modified	515304	774 ×779	-223.32	67.10%	28.37%	4.53%
bigblue2_modified	576816	468×471	-443.19	63.64%	31.52%	4.85%
newblue1_modified	331663	399×399	-163.74	60.43%	30.29%	9.27%
newblue3_modified	551667	973×1256	-117.67	61.53%	24.80%	9.66%
newblue4_modified	636195	455×458	-318.46	62.63%	32.95%	4.42%
newblue5_modified	1257555	637×640	-38.97	60.94%	31.62%	7.44%
Average	563400		-198.41	64.49%	28.51%	7.00%

In Table 1, the first column is the name of each ISPD2008 modified benchmark, the second and third columns show the number of nets and grids in each benchmark, the forth column shows the worst negative slack of the nets in each benchmark. Columns five to eight show the average wire density on 1x, 2x and 4x metal layers.

Table 2. ISPD2008 adaptec2 modified benchmarks comparison

Adaptec2	**Conventional Flow**	**Layer Aware Design Flow**	**Improvement**
CPU time(s)	175695	19391.2	
Number of buffer	557110	307929	44.73%
Buffer capacitance(pF)	2885.64	1374.8	52.36%
Wire area(μm^2)	4.54255e+7	4.52451e+7	0.3988%
Wire capacitance(pF)	4.78216e+5	4.76309e+5	0.3988%
Overflow	0	0	
Average slack(ps)	253.932	162.203	

Table 3. ISPD2008 bigblue2 modified benchmarks comparison

Bigblue2	**Conventional Flow**	**Layer Aware Design Flow**	**Improvement**
CPU time(s)	22791.7	182771	
Number of buffer	650164	475794	26.82%
Buffer capacitance(pF)	3551.65	2194.23	38.22%
Wire area(μm^2)	1.13038e+8	1.11826e+8	0.1084%
Wire capacitance(pF)	1.08904e+6	1.08786e+6	0.1084%
Overflow	0	0	
Average slack(ps)	174.645	160.472	

Table 4. ISPD2008 newblue1 modified benchmarks comparison

newblue1	**Conventional Flow**	**Layer Aware Design Flow**	**Improvement**
CPU time(s)	26639.6	398431	
Number of buffer	688887	375994	45.42%
Buffer capacitance(pF)	3805.96	1847.93	51.45%
Wire area(μm^2)	5.32458e+7	5.31033e+7	0.2683%
Wire capacitance(pF)	7.23058e+5	7.21118e+5	0.2683%
Overflow	102	0	
Average slack(ps)	204.665	153.243	

Table 5. ISPD2008 newblue3 modified benchmarks comparison

newblue3	**Conventional Flow**	**Layer Aware Design Flow**	**Improvement**
CPU time(s)	30578.8	205999	
Number of buffer	713141	490747	31.19%
Buffer capacitance(pF)	3656.77	2114.06	42.19%
Wire area(μm^2)	1.28366e+6	1.28279e+8	0.06787%
Wire capacitance(pF)	1.59363e+6	1.59255e+6	0.06787%
Overflow	0	0	
Average slack(ps)	282.026	208.467	

Table 6. ISPD2008 newblue4 modified benchmarks comparison

newblue4	**Conventional Flow**	**Layer Aware Design Flow**	**Improvement**
CPU time(s)	46177.7	379744	
Number of buffer	1351093	828330	38.69%
Buffer capacitance(pF)	6976.63	3577.89	48.72%
Wire area(μm^2)	1.75580e+8	1.75491e+8	0.051%
Wire capacitance(pF)	2.12145e+6	2.12038e+6	0.051%
Overflow	892	0	
Average slack(ps)	333.623	236.280	

Table 7. ISPD2008 newblue5 modified benchmarks comparison

newblue5	**Conventional Flow**	**Layer Aware Design Flow**	**Improvement**
CPU time(s)	71182.5	742135	
Number of buffer	1695839	1072247	36.77%
Buffer capacitance(pF)	8830.80	4720.54	46.54%
Wire area(μm^2)	3.50300e+8	3.49847e+8	0.1293%
Wire capacitance(pF)	4.94064e+6	4.93424e+6	0.1293%
Overflow	0	0	
Average slack(ps)	315.220	232.431	

Table 8. ISPD2008 adaptec3 modified benchmarks comparison

Adaptec3	**Conventional Flow**	**Layer Aware Design Flow**	**Improvement**
CPU time(s)	49140.3	332257	
Number of buffer	688821	406382	41.00%
Buffer capacitance(pF)	3593.17	1823.16	49.26%
Wire area(μm^2)	1.22985e+8	1.22474e+8	0.4156%
Wire capacitance(pF)	1.52307e+6	1.51674e+6	0.4156%
Overflow	0	0	
Average slack(ps)	220.309	148.637	

Table 9. ISPD2008 adaptec4 modified benchmarks comparison

Adaptec4	**Conventional Flow**	**Layer Aware Design Flow**	**Improvement**
CPU time(s)	30407.3	243837	
Number of buffer	735331	490794	33.26%
Buffer capacitance(pF)	3851.23	2226.73	42.18%
Wire area(μm^2)	1.29747e+8	1.29239e+6	0.3915%
Wire capacitance(pF)	1.69367e+6	1.68704e+6	0.3915%
Overflow	0	0	
Average slack(ps)	217.464	210.800	

In the above tables, the second rows show the CPU time. Row three to six show the buffer and wire result, the usage of buffer number and capacitance, and the usage of wire area and capacitance, respectively. The seventh rows show the overflow of this benchmark. The last row shows the average slack of this benchmark. In modified benchmark ISPD2008-adaptec2, when set constant weight factors $\alpha = 100$ and $\beta = 1$, our flow reduces buffer capacitance by 52.36% and reduces wire capacitance by 0.40%. By comparison, when set $\alpha = 10$ and $\beta = 1$, our flow reduces buffer capacitance by 13.4% and reduces wire capacitance by 0.57%. In tables 2-9, α is set as 100 and β is set as 1.

According to the results, Tables 2-9 show that our flow reduces the number of buffers by 37.24% on average. The average reduction of buffer capacitance is 46.37%. Our flow also reduces wire area and capacitance by 0.23% on average. For example, in modified benchmark ISPD2008-newblue4, although our layer centric design flow runs about 105 hours and traditional flow runs only about 13 hours, our flow reduces the wire area and capacitance by 0.051%, especially, our layer centric flow reduces the number of buffers by 38.69% and reduces buffer capacitance by 48.72%. In the meantime, after global routing, there is no overflow based on the simultaneous buffering and layer assignment with layer directive guidance. However, the modified benchmark ISPD2008-newblue4 still have 892 overflow by conventional design flow.

In addition, as shown in table 2, 3, 5, 7, 8, 9, both conventional flow and our flow have no overflow, but the buffer and wire result still achieved around 35.63% buffer number, 45.13% buffer capacitance and 0.25% wire area and capacitance on average.

Overall, our layer centric design flow is proved to reduce the buffer and wire resources and achieve good solutions for better global routing.

6. CONCLUSIONS

In this thesis, we proposed FLUTE algorithm for Rectilinear Steiner Minimal Tree (RSMT) construction; nonlinear conjugate gradient algorithm for the preliminary layer planning; Lagrangian Relaxation algorithm for buffer insertion and layer assignment and NTHU-router for the global routing. Our design flow reduce the buffer and metal resources usage while meeting the timing constraints. Meanwhile, our flow reduces the overflow for some benchmarks. For commercial application, the reduction of buffer and metal resources usage can significantly decrease the design power and cost, and improve the design closure.

Compared with the traditional physical design flow, our approach achieved around 37.24% buffer number, 46.37% buffer capacitance and about 0.23% wire area & capacitance on average.

REFERENCES

[1] S. Hu, Z. Li, and C. J. Alpert, "A polynomial time approximation scheme for timing constrained minimum cost layer assignment," in *Proceedings of* the 2008 IEEE/ACM International Conference on Computer-Aided Design, November 10-13, 2008, San Jose, California, USA.

[2] Y.-J. Chang, Y.-T. Lee, and 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.

[3] C. Chu, "FLUTE: fast lookup table based wirelength estimation technique," in *Proceedings of* the 2004 IEEE/ACM International Conference on Computer-Aided Design, November 7-11, 2004, San Jose, California, USA.

[4] Y. Wei, Z. Li, C. Sze, S. Hu, C. J. Alpert, and S. S. Sapatnekar, "CATALYST: Planning layer directives for effective design closure," in *Proceedings of* the Design Automation & Test in Europe Conference & Exhibition, March 18-22, 2013, Grenoble, France.

[5] J.-R. Gao, P.-C. Wu, and T.-C. Wang, "A new global router for modern designs," in *Proceedings of* the 2008 Asia and South Pacific Design Automation Conference, March 21-24, 2008, Seoul, Korea.

[6] J. A. Roy and I. L. Markov, "High-performance routing at the nanometer scale," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 27, no. 6, pp. 1066-1077, 2008.

[7] P. Saxena and C. L. Liu, "Optimization of the maximum delay of global interconnects during layer assignment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 4, pp. 503-515, 2001.

[8] Y. Jia, Y. Cai, and X. Hong, "Timing Driven Layer Assignment Considering Via Resistance and Coupling Capacitance," in *Proceedings of* the International Conference on Communications, Circuits and Systems, July 11-13, 2007, Kokura, Japan.

[9] C. –Y. Chang (2012). Analytical Layer Planning for Nanometer VLSI Designs. Master's thesis, Texas A&M University.

[10] J. -T. Tsai (2013). VLSI Interconnect Optimization Considering Non-uniform Metal Stacks. Master's thesis, Texas A & M University.