VLSI INTERCONNECT OPTIMIZATION CONSIDERING NON-UNIFORM METAL STACKS

A Thesis

by

JUNG-TAI TSAI

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

Chair of Committee,	Jiang Hu
Committee Members,	Weiping Shi
	Rabi N. Mahapatra
Head of Department,	Chanan Singh

August 2013

Major Subject: Computer Engineering

ABSTRACT

With the advances in process technology, comes the domination of interconnect in the overall propagation delay in modern VLSI designs. Hence, interconnect synthesis techniques, such as buffer insertion, wire sizing and layer assignment play critical roles in the successful timing closure for EDA tools. In this thesis, while our aim is to satisfy timing constraints, accounting for the overhead caused by these optimization techniques is of another primary concern.

We utilized a Lagrangian relaxation method to minimize the usage of buffers and metal resources to meet the timing constraints. Compared with the previous work that extended traditional Van Ginneken's algorithm, which allows for bumping up the wire from thin to thick given significant delay improvement, our approach achieved around 25% reduction in buffer + wire capacitance under the same timing budget.

To my family

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Dr. Jiang Hu. The completion of this thesis would not be possible without his patient guidance and helpful feedback. His encouragement and positive influence has made my years in TAMU productive and enriching. In addition, much appreciation is given to Dr. Weiping Shi and Dr. Rabi Mahapatra for spending your precious time coaching me and imparting your valuable experiences.

Secondly, I want to thank my fellow group members for being my alliance and support through the tough times and good times. You all have been great influential forces, be it in my academic career or life.

Lastly, I want to thank my dearest family for their unconditional support and nurturance. They have given me the courage and motivation to defeat all obstacles in the pursuit of this degree.

This thesis is hereby dedicated for each and every one of you, who love and care for me. Thank you for all that you do.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation

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 typical design flows, many interconnect optimization techniques are performed between the placement and routing, buffer insertion is the most efficient way of them to improve the design performance.

As new nanometer process technology with multiple routing layers is becoming available [1] and metal parasitic among different layers becoming increasingly non-uniform, layer and its corresponding parasitic information are necessary in these optimization techniques. Without layer and corresponding parasitic information, the interconnect optimizations may unnecessarily insert huge amount of buffers so that more power is consumed and more areas of chip are occupied.

Moreover, besides considering the issues of wirelength and overflow, the modern router also takes into account the detours of timing-critical nets into high layer. Although assigning wires to thick metal results in efficient timing, routability of the design may be compromised. Thus, good use of thicker metal is our other target.

1.2. Buffer Insertion

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 strategy is often widely used in modern VLSI designs to achieve better performance [2], [3], [4], [5], [6], [7], [8], and [9].

However, these inserted buffers consume more power and occupy more areas of chip. Hence, recent research emphasizes more on the arrangement of buffer resources, which is different from the traditional Van Ginneken's algorithm [7] that only aggressively improves interconnect timing. Having taken into consideration of power consumption, [2], Lillis et al. [3], Liu et al. [4], and Peng and Liu [9] proposed techniques to minimize the total buffer size while satisfying a given timing constraint for power minimization.

1.3. Layer Assignment

As the shift towards nanoscale becomes a prevailing trend among VLSI technology, interconnect delay is identified as a barrier in circuit performance due to the increasingly resistive wires that make transmitting of signal across the chip difficult.

However, as many more routing layers are becoming available with some 65 nm technologies consisting of eight layers of metal, and certain 45 nm technologies having ten layers [10], layer assignment has become a new optimization techniques for

achieving efficient timing closure. Wires found on the upper metal layers have much less resistivity, and can be used to drive further and more efficiently than on thinner metals.

Although assigning all wires to thick metal results in efficient timing, routability of the design may be compromised. Thus, good use of thicker metal is our other target. Our aim is to bump up wires that really need timing improvement to thicker layer, so that router can have sufficiently thicker wire resources for processing other more critical nets. The challenge in this process is the assigning of minimal amount of wires to thick metal for meeting timing constraints.

1.4. Our Contributions

The traditional Van Ginneken's based algorithms often tries to achieve the minimum possible interconnect delay, and thus causing unnecessary waste of power and limited thick metal resources. To alleviate this problem, we proposed an alternative approach to minimize the usage of buffers and thick layer while efficiently utilize the interconnect timing budget.

In this thesis, we adopt a Lagrangian relaxation method to minimize the usage of buffers and metal resources while meeting the timing constraints. Compared with the previous work [1] that extended traditional Van Ginneken's algorithm that allows for the bumping up of wires from thin to thick given significant delay improvement, our approach achieved around 25% reduction in buffer + wire capacitance.

1.5. Overview

The remainder of this thesis is organized as follows. Section 2 briefly introduces the backgound information and formulates the problem. Section 3 presents our multi-objective models that simultaneously minimize buffer and matel capacitance. Senction 4 shows our experimental results. Section 5 concludes this thesis.

2. PROBLEM FORMULATION

2.1. Circuit Model

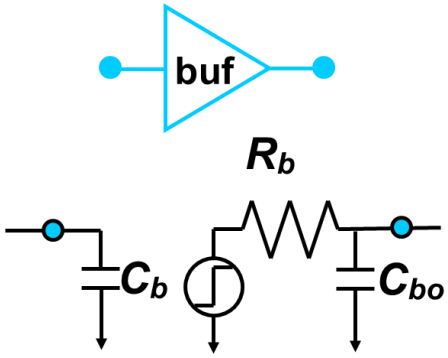

Fig. 1(a) Switch-level RC model for buffer

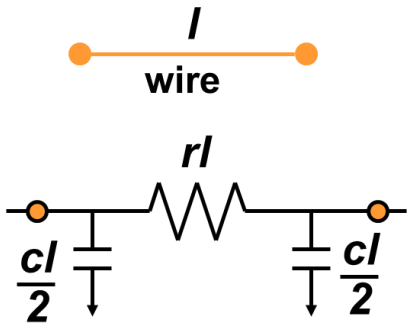

Fig. 1(b) Lumped RC π-model for interconnect

For this experiment the buffers are modeled after switch level RC model, in which C_b is the input capacitance seen by the fan-in signal, R_b and C_{bo} are the output resistance and the output capacitance of the buffer. The circuit model is shown in Fig. 1(a).

The lumped RC model shown in Fig. 1(b) is then used to model each uniform interconnect. l denotes the interconnect length, r and c are the unit resistance and unit capacitance of the interconnect. To compute the delay of the circuit, the widely adopted Elmore delay model [2, 3, 4] is used.

The total buffer capacitance will be used as the measure of power dissipation employed in [11]. The reason being that dynamic and leakage power dissipation of buffer are proportional to its capacitance. For the rest of this thesis, we replace the power minimization by the minimization of the total buffer capacitance.

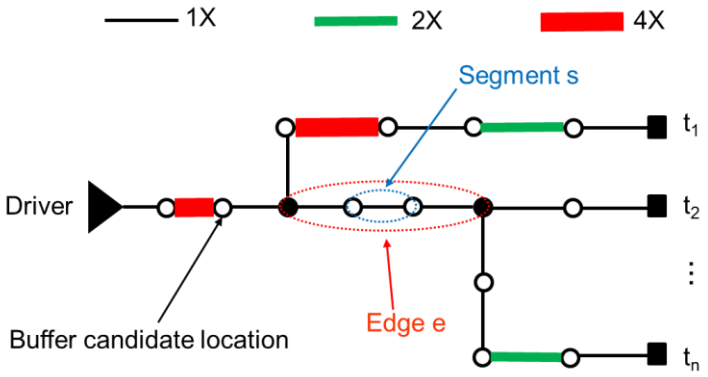

Fig. 2. Multi-layer interconnect tree

2.2. Multi-layer Interconnect Tree

Fig. 2 shows the structure of a multi-layer interconnect tree $T = (V, E)$ where V is a set of nodes and E is a set of edges. Also, each edge e is made of several segment s. Each segment s is assumed to be routed on any specific layer within the multiple layer choices. In Fig. 2, there are three layer thickness options, 1x, 2x and 4x.

Supposed that a set of M routing layers $l_1, l_2, \ldots, l_M$ are given where l_1 is the thinnest layer and l_M is the thickest layer. Each layer l_i is associated with its own unit resistance and unit capacitance. In Fig. 3, the length l_s, unit resistance r_s and unit capacitance c_s for segment s are the corresponding parameter values of layer l_i when the segment s is assigned to layer l_i.

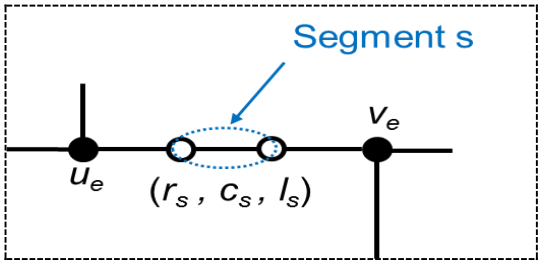

Fig. 3. An edge and a segment

The output resistance R_s of the driver *src* is given. For each sink t_i ($i = 1, ..., n$), the required arrival time and downstream capacitance are given. Also, the number of buffer candidate locations of each edge have been determined manually. Then, simultaneous buffer insertion and layer assignment on each edge are ready to be performed to minimize the total buffer capacitance and metal capacitance under the timing budget.

The formulations for this multi-objective buffer insertion and layer assignment problem are proposed.

2.3. Primal Problem

In Fig. 3, the beginning and end points of edge e are denoted as u_e and v_e. X_e denotes the total buffer capacitance inserted on edge e, and Y_e denotes the total metal capacitance used for edge e. Let d_e be the signal delay of edge e. Let a_k be the *required arrival time (RAT)* at node $k \in V$, and q_i be the RAT for sink t_i. We combine the buffer and metal

capacitance into one objective function, expressed as $\sum_{e \in E} \alpha X_e + \beta Y_e$. Then the primal problem of buffer and metal optimization is formulated as follows:

$$Min \quad \sum_{e \in E} \alpha X_e + \beta Y_e \tag{1}$$

$$s.t. \quad a_{src} \geq 0 \tag{2}$$

$$a_{ue} + d_e \leq a_{ve} \tag{3}$$

$$\forall v_e \in sinks, \; a_{ve} = q_i \tag{4}$$

Where α and β denote the weights of the objective.

3. OUR APPROACH

A large number of inequality constraints exist in the primal problem. Moreover, unknowns a_k and the solutions of buffer insertion and layer assignment have yet to be calculated. Thus, the primal problem is translated into a dual problem using Lagrangian relaxation method for the relief of complex nature [9], [12], [13], [14], [15].

In Lagrangian relaxation method, constraints are incorporated into the objective function called *Lagrangian relaxation function* by introducing non-negative Lagrangian multipliers. Then the dual problem can be solved by heuristic techniques.

3.1. Lagrangian Relaxation

The Lagrangian relaxation function is written as:

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

where $\vec{X}$ is the vector of total buffer capacitance of $X_e, \forall e \in E$, $\vec{Y}$ is the vector of total metal capacitance of $Y_e, \forall e \in E$, $\vec{a}$ are the RATs of all nodes, $\vec{\lambda}$ is the Lagrangian multiplier vector in which λ_{src} is the Lagrangian multiplier for constraint (2), λ_e is the Lagrangian multipliers of edge e for constraint (3), and $a_{ve} = q_i$ for $v_e = t_i$.

According to Karush-Kuhn-Tucker (KKT) Theorem, at the optimal solution, $\frac{\partial L(\vec{X},\vec{Y},\vec{a},\vec{\lambda})}{\partial a_k} = 0$ for all node *k*. From this condition, we can derive the equation:

$$\lambda_e = \sum_{\hat{e} \in des(e)} \lambda_{\hat{e}} \qquad (6)$$

where *des(e)* denotes the set of descendant edges of edge *e*.

By equation (6), we can eliminate the terms with RAT at all nodes while keeping the terms with RAT at sinks, to reduce equation (5) into a simpler form (7). In addition, after the Lagrangian multipliers are fixed in each iteration, $L(\vec{X},\vec{Y},\vec{a},\vec{\lambda})$ can be rewritten as $L_{\vec{\lambda}}(\vec{X},\vec{Y})$.

$$\begin{aligned} L(\vec{X},\vec{Y},\vec{a},\vec{\lambda}) &= \sum_{e\in E} \alpha X_e + \beta Y_e + \sum_{e\in E} \lambda_e (a_{ue} + d_e - a_{ve}) - \lambda_{src} a_{src} \\ &= \sum_{e\in E} \alpha X_e + \beta Y_e + \sum_{e\in E} \lambda_e d_e + \sum_{e\in E} \lambda_e (a_{ue} - a_{ve}) - \lambda_{src} a_{src} \\ &= \sum_{e\in E} \alpha X_e + \beta Y_e + \sum_{e\in E} \lambda_e d_e + \sum_{e\in E} (\lambda_e a_{ue} - \sum_{\hat{e}\in des(e)} \lambda_{\hat{e}} a_{ve}) - \lambda_{src} a_{src} \\ &= \sum_{e\in E} (\alpha X_e + \beta Y_e + \lambda_e d_e) - \sum_{\forall ve \in ti} \lambda_e q_i \qquad (7) \end{aligned}$$

Then, the Lagrangian relaxation dual of the primal problem of simultaneous buffer insertion and layer assignment is described as follows:

$$\begin{aligned} &Max \quad Q_{\vec{\lambda}} \\ &\quad s.t. \qquad \lambda_e \geq 0, \forall e \in E \\ &\qquad\qquad \lambda_e = \sum_{\hat{e}\in des(e)} \lambda_{\hat{e}} \\ &where\ Q_{\vec{\lambda}} = min\ L_{\vec{\lambda}}(\vec{X},\vec{Y}) \qquad (8) \end{aligned}$$

In equation (8), $Q_{\vec{\lambda}}$ is defined as the minimal value of $L_{\vec{\lambda}}(\vec{X},\vec{Y})$. Given the Lagrangian multiplier vector in each iteration, the Lagrangian functions (7) can be solved heuristically using dynamic programming.

3.2. Heuristic Solver

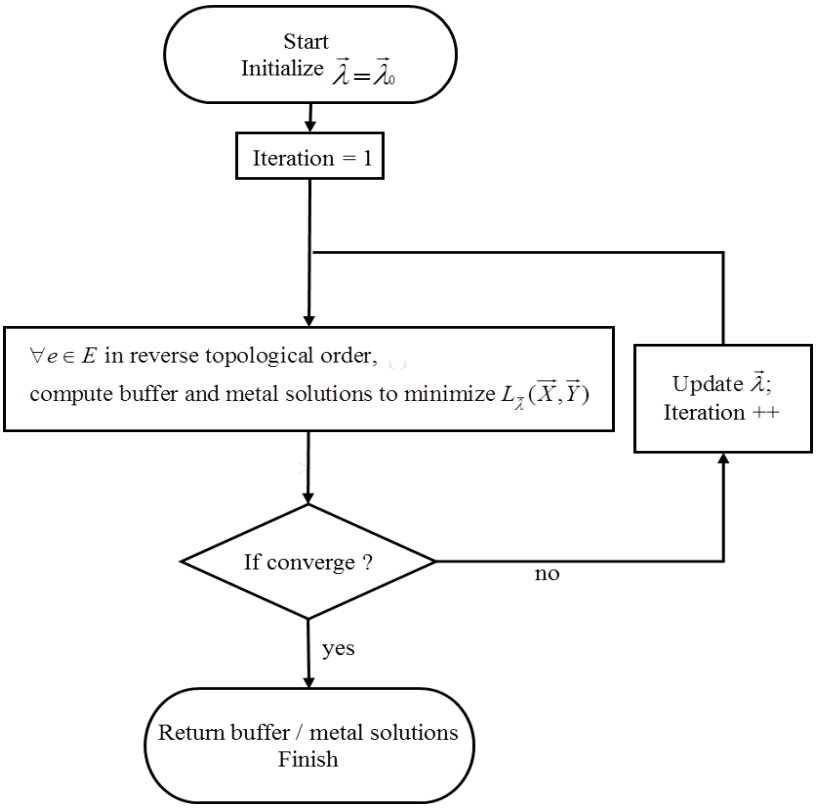

Fig. 4. The flow chart of solving the Lagrange dual problem

Fig.4 shows the flow chart of algorithm used to solve the Lagrange dual problem. First, the Lagrangian multipliers was initialized. Under the given multiplier values during each iteration, the edges are processed in reverse topological order.

During the current iteration with a fixed $\vec{\lambda}$, we insert buffers and assign segments of edges to the most appropriate layers to minimize the contribution to the value of $L_{\vec{\lambda}}(\overrightarrow{X},\overrightarrow{Y})$. At the end of the process in this iteration, we can select the best solution at the source node to determine the minimal value of $L_{\vec{\lambda}}(\overrightarrow{X},\overrightarrow{Y})$.

According to the results from the current iteration, a new $\vec{\lambda}$ is derived for the next iteration. This process is repeated until convergence occurs under a fixed threshold, afterwards the solutions in the current iteration can be returned.

The convergence condition can be set as $\frac{Q_{\vec{\lambda}}^{(i+1)} - Q_{\vec{\lambda}}^{(i)}}{Q_{\vec{\lambda}}^{(i)}} < \varepsilon$ [9], [15], where $Q_{\vec{\lambda}}^{(i)}$ is the value of $Q_{\vec{\lambda}}$ at the iteration *i*, and ε is a user defined threshold.

3.2.1. Local Optimizer

The first main component in solving the Lagrangian dual problem is to use dynamic programming algorithm as shown in Fig. 5 to minimize $L_{\vec{\lambda}}(\overrightarrow{X},\overrightarrow{Y})$, under a given $\vec{\lambda}$ during each iteration. The inputs of this pseudocode consist of a binary routing tree *T*, buffer library *B*, *M* leyers, and a given $\vec{\lambda}$. After processing an entire interconnect tree in reverse topological order, the solutions of buffer insertion and layer assignment will be derived, and the minimal value of $L_{\vec{\lambda}}(\overrightarrow{X},\overrightarrow{Y})$, namely $Q_{\vec{\lambda}}$, will be returned.

For an entire binary interconnect tree, each buffer candidate location has a set of non-inferior solutions. Each solution is denoted by a 2-tuple (*obj*, *cap*), where *obj*

1. for each edge e in reverse topological order
2. if $v_e \in sinks$
3. $S_{ve} = \{(0, c_{ti})\}$
4. else if e has an only descendant $\hat{e}$
5. $S_{ve} = S_{u\hat{e}}$
6. else /* e has two descendants eL, eR */
7. for any $(obj_1, cap_1) \in S_{u(eL)}$ and $(obj_2, cap_2) \in S_{u(eR)}$
8. $sol = (obj_1 + obj_2, cap_1 + cap_2)$
9. $S_{ve} = S_{ve} \cup \{sol\}$;
10. prune S_{ve}
13. /* Assume e has k buffer candidate locations */
14. for $i = 1$ to k /* buffer candidate locations $i = 1$ is the closest to node v_e, $S_{e,i=0} = S_{ve}$*/
15. for each $(obj, cap) \in S_{e,i-1}$
16. for each metal type $m \in M$
17. $l_s = l_m$; $c_s = c_m$; $r_s = r_m$;
18. for each buffer type $b \in B$
19. if $b = 0$
20. $sol = (obj + \lambda_e r_s l_s(\frac{c_s l_s}{2} + cap),\ cap + c_s l_s)$
21. else
22. $sol = (obj + \lambda_e(R_b(c_s l_s + cap + C_{bo}) + r_s l_s(\frac{c_s l_s}{2} + cap)),\ C_b)$
23. $S_{e,i} = S_{e,i} \cup \{sol\}$
24. prune $S_{e,i}$
25. for each $(obj, cap) \in S_{e,k}$
26. for each metal type $m \in M$
27. $l_s = l_m$; $c_s = c_m$; $r_s = r_m$;
28. $sol = (obj + \lambda_e r_s l_s(\frac{c_s l_s}{2} + cap),\ cap + c_s l_s)$
29. $S_{ue} = S_{ue} \cup \{sol\}$
30. prune S_{ue}
31. Find the tuple (obj^*, cap^*) at the interconnect root with minimal obj

Fig. 5. Dynamic programming scheme

represents the total contribution to the value of $L_{\vec{\lambda}}(\overrightarrow{X},\overrightarrow{Y})$ from all the downstream candidate locations including current candidate location. *cap* denotes the downstream capacitance at current candidate location.

For each edge *e*, the solution set associated with its endpoint v_e is derived first in lines 2 – 10. If $v_e \in sinks$, $S_{v_e} = \{(0, c_{t_i})\}$, where c_{t_i} is the given sink capacitance of t_i. If *e* has only one descendant $\hat{e}$, because v_e and $u_{\hat{e}}$ are the same vertex, $S_{v_e} = S_{u_{\hat{e}}}$. If *e* has two descendants *eL* and *eR*, S_{v_e} is derived by merging any pair of solutions, one from $S_{u(eL)}$ and one from $S_{u(eR)}$. It is found that a solution (*obj1, cap1*) is inferior to (*obj2, cap2*) if *cap1 > cap2* and *obj1 > obj2*. Inferior solutions are then pruned. Slew constraint [16] is also considered in the pruning procedure to efficiently reduce the size of solution set while keeping the solution quality.

Assume that there are *k* buffer candidate locations along the edge *e*. After the derivation of S_{v_e}, in lines 14-24, edge *e* is processed with its buffer candidate locations starting from the one closest to its end point v_e. Here $S_{e,i}$ denotes the solution set associated at the candidate location *i (i = 1, ..., k)*, along edge *e*. S_{v_e} and S_{u_e} can be seen as $S_{e,0}$ and $S_{e,k+1}$, respectively.

In lines 16-17, l_s is the length of the segment *s* of edge *e*. Unit capacitance c_s and unit resistance r_s for segment *s* will be assigned the corresponding parameter values c_m and r_m of layer m, when the segment *s* is selected to use layer *m*.

In lines 18-24, for each candidate location along the edge *e*, all buffer options (including no buffer inserted) are analyzed and the corresponding set of non-inferior

solutions is computed.

In Lines 25–30, S_{ue} is derived in a similar fashion. At the end of the process, we can select the solution with minimum *obj* at the root to determine the minimal value of $L_{\vec{\lambda}}(\vec{X},\vec{Y})$ at Line 31.

3.2.2. Update of Lagrangian Multiplier

An efficient update scheme of Lagrangian multipliers will speed up the run time in solving the Lagrangian dual problem. Useful update scheme is crucial to increase the convergence rate so that our approach can converge faster due to less iterations.

The convergence rate of the traditional subgradient method is unstable because it is extremely sensitive to not only the step size but also the initial values of the Lagrangian multipliers [9]. Hence, after each current iteration, a more efficient mathematical scheme called the ellipsoid method is used to compute the new Lagrangia multipliers for the next iteration.

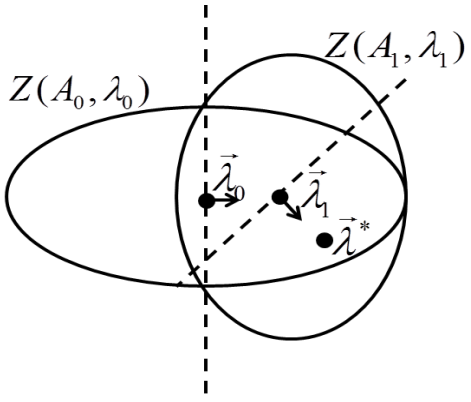

Fig. 6. Illustration of ellipsois method

The basic principle of ellipsoid method is illustrated in Fig. 6. We use $\vec{\lambda}$ to denote the Lagrange multipliers $\lambda_e's$ of the edges which are connected to sinks $v_e = t_i$.

We first initialize $\vec{\lambda}$ and A as the center vector and volume matrix of a ellipsoid $Z(A_0, \vec{\lambda}_0) = \{\vec{z} \mid (\vec{z} - \vec{\lambda}_0)^T A_0^{-1} (\vec{z} - \vec{\lambda}_0) \leq 1\}$ at iteration 0, which contains the optimal solution $\vec{\lambda}^*$ with maximum $Q_{\vec{\lambda}}$. We initialize A to be a diagonal matrix with the element $a_{ii} = \lambda_{\max i}^2, i \in \{0,1,\cdots, |\vec{\lambda}| - 1\}$, where $\lambda_{\max i}$ is an initial Lagrangian multiplier related to the sink t_i.

Then, we use the subgradient of $Q_{\vec{\lambda}}$ at $\vec{\lambda}_0$ to generate a cut plane passing through $\vec{\lambda}_0$, which divides the $Z(A_0, \vec{\lambda}_0)$ into two parts, with the optimal solution $\vec{\lambda}^*$ included in one of the parts. A new ellipsoid $Z(A_1, \vec{\lambda}_1)$ with smaller volume is then created to cover the partition that contains the optimum. This procedure is conducted iteratively until the volume of the ellipsoid containing the optimum is smaller than the predetermined threshold. Subsequently, the optimal solution will be estimated as the ellipsoid center.

The steps in the Fig. 7 compute the new $(\vec{\lambda}, A)$ pair to represent the new ellipsoid. In the beginning, the timing slack set $\vec{S}$ for *n* sinks are computed. In line 2, a vector $\vec{g}$ is derived by the values of $\vec{S}$ and A. Then the new pair $(\vec{\lambda}, A)$ can be created in lines 3-4.

During each iteration, Lagrange multipliers $\lambda_e's$ of the edges, which are connected to sinks $v_e = t_i$, will be updated by ellipsoid method. According to equation (6), the Lagrange multipliers $\lambda_e's$ of other edges are then updated by the edges connected to the sinks.

Ellipsoid method($\vec{\lambda}, A$)

1. timing slack vector $\vec{S}$ for n sinks are computed

2. $\vec{g} = \vec{S} / \sqrt{\vec{S}^T A \vec{S}}$

3. $\vec{\lambda} = \vec{\lambda} - \frac{1}{|\vec{\lambda}| + 1} A\vec{g}$

4. $A = \frac{|\vec{\lambda}|^2}{|\vec{\lambda}|^2 - 1}(A - \frac{2}{|\vec{\lambda}| + 1} A\vec{g}\vec{g}^T A)$

5. return ($\vec{\lambda}$, A)

Fig. 7. Ellipsois method

4. EXPERIMENT RESULTS

We conduct the experiments on a set of interconnect trees. The number of sinks for the interconnect trees ranges from 5 to 25. The length of each wire for the trees ranges from 1000 to 12000μm. We further divided each edge into *k* uniformed divisions (*k* = 20) to determine the buffer candidate locations of each edge. Then we set α = 100, and β = 1 as the value of the weights in the objective function $\sum_{e \in E} \alpha X_e + \beta Y_e$.

The timing constraint of each interconnect is set from $0.5\,\tau_{\max}$ to $0.9\,\tau_{\max}$ where $\tau_{\max}$ is the timing when only the *1x* layer and w/o buffer options are used. Also, we set another timing constraint, $\gamma\sqrt{\tau_{\max}}$, $\gamma = 30$ to give a more restrictive timing budget for each interconnect testcase.

The initial Lagrange multipliers $\lambda_e's$ of edges, which are connected to sinks $v_e = t_i$, are set to 0.1. According to equation (6), the Lagrange multipliers $\lambda_e's$ of other edges are completely determined by those of the edges connected to the sinks. The value of ε is set to 0.1%.

We then compare the results to previous work [1], which extends the classic van Ginneken/Lillis algorithm [3], [7] and directly controls the wire resources by timing improvement. The foundamental idea of this previous work is that during interconnect buffering, the segment of an edge receives a raise from current thinner layer to the next thicker layer when the assignment caused a significant delay improvement. Therefore,

assigning a wire to thicker layer is directly controlled by the timing improvement before/after layer assignment and the threshold of the current layer. Both [1] and our approach are implemented by C++ under Linux operating system.

Table 1. Comparison between previous work [1] and our complete approach under timing constraint = $\mathbf{0.5}\,\tau_{\max}$

Timing Constraint $= 0.5\,\tau_{\max}$		LADY					Ours				
Net	Sinks	Slack(ps)	Buffer Cap(pf)	Wire Cap(pf)	Total Cap(pf)	Time(sec.)	Slack(ps)	Buffer Cap(pf)	Wire Cap(pf)	Total Cap(pf)	Time(sec.)
Net5-1	5	3.80835	0.087282	6.647619	6.734901	0.46	562.518	0.039471	5.257142	5.296613	1.45
Net5-2	5	3.80835	0.094539	6.4	6.494539	0.45	562.518	0.046471	5.214285	5.260756	1.45
Net6-1	6	4830.82	0.12128	6.195238	6.316518	1.37	448.872	0.039487	4.95238	4.991867	14.43
Net6-2	6	5989.67	0.150308	7.394285	7.544593	1.36	317.123	0.044639	6.182857	6.227496	9.24
Net8	8	3167.58	0.121312	4.714285	4.835597	2.5	2343.75	0.04638	4	4.04638	13.65
Net10	10	1515.45	0.133218	6.861904	6.995122	4.91	1149.92	0.05112	4.921428	4.972548	49.24
Net14	14	6221.46	0.213334	12.523809	12.737143	13.26	92.8548	0.062125	8.566666	8.628791	9.77
Net17	17	3535.81	0.213564	9.747619	9.961183	24.19	798.352	0.071305	7.433333	7.504638	64.78
Net20	20	2605.89	0.320405	14.290476	14.610881	39.62	1891.46	0.101157	10.519047	10.620204	59.28
Net25	**25**	**3840**	**0.375719**	**13.728571**	**14.10429**	**78.56**	**964.105**	**0.100531**	**11.37619**	**11.476721**	**52.2**
Average		3171.43	0.1830961	8.8503806	9.0334767	16.668	913.1473	0.0602686	6.8423328	6.9026014	27.549
Normalization		1	1	1	1	1	0.287929	0.3291637	0.7731117	**0.7641134**	1.6528077

Table 1 shows the comparison among the five items: slacks, buffer capacitance, wire capacitance, total capacitance and CPU time between [1] and our approach while meeting the timing budget $0.5\,\tau_{\max}$.

In Table 1, the first two columns show the name of each interconnect tree, and the number of sinks. Columns eight to twelve show the slacks, the usage of buffer capacitance, the usage of wire capacitance, the total capacitance and the required CPU time in our approach, respectively. We also apply the algorithm of [1] to these 10 interconnect trees to obtain the values of these five items for comparison and record them in columns three to seven, respectively.

Take Net25 as an example, after running 52.2 seconds, our approach terminates and achieves positive slack at 562.52 ps. At the same time, our approach uses buffer, wire and total resources at 0.039 pf, 5.257 pf and 5.297 pf, respectively. In constrast, [1] uses more buffer, wire and total resources at 0.087 pf, 6.648 pf and 6.735 pf, respectively and improves less timing, (i.e., the slack is 3.80ps). However, [1] requires less CPU time (0.46 seconds) than our method because many more iterations were used to find optimal result in our approach.

Then we take the average among the 10 trials as shown in the second to the last row in Table 1. Moreover, we obtained values of normalization by dividing our obtained average by the corresponding average values in [1]. The experimental results show that compared with [1], our approach achieved around 23.6% reduction in buffer + wire capacitance on average.

Table 2 shows the comparison on inserted buffer numbers and metal usage distribution between [1] and our approach while meeting the timing budget $0.5\tau_{\max}$. In Table 2, the first two columns show the name of each interconnect tree, and the number of sinks. Columns seven to ten show the buffer numbers, the percentage of use on metal

1x, 2x and 4x out of total use in our approach. We also apply the algorithm of [1] to these 10 interconnect trees to obtain the values of these items for comparison and record them in columns three to six, respectively.

Table 2. Comparison on buffer numbers and layer distribution between previous work [1] and our complete approach under timing constraint = $\mathbf{0.5}\,\tau_{\max}$

Timing Constraint $= 0.5\,\tau_{\max}$		LADY				Ours			
Net	Sinks	Buffer No.	Layer Distribution			Buffer No.	Layer Distribution		
			1x(%)	2x(%)	4x(%)		1x(%)	2x(%)	4x(%)
Net5-1	5	14	0.585921325	0.398550725	0.015528	13	0.890269151	0.093167702	0.01656315
Net5-2	5	15	0.522702104	0.471760797	0.0055371	11	0.838316722	0.136212625	0.02547065
Net6-1	6	20	0.722391084	0.257345491	0.0202634	13	0.946301925	0.053698075	0
Net6-2	6	24	0.775793651	0.194444444	0.0297619	17	0.932539683	0.064484127	0.00297619
Net8	8	20	0.789473684	0.195488722	0.0150376	10	0.947368421	0.052631579	0
Net10	10	20	0.577235772	0.297328688	0.1254355	10	0.832171893	0.151567944	0.01626016
Net14	14	34	0.621916236	0.312679289	0.0654045	25	0.971313827	0.026965003	0.00172117
Net17	17	36	0.675385647	0.300469484	0.0241449	21	0.955734406	0.042924212	0.00134138
Net20	20	55	0.681428571	0.263333333	0.0552381	33	0.9519047619047	0.046190476	0.00190476
Net25	**25**	**59**	**0.71474359**	**0.267857143**	**0.0173993**	**27**	**0.913461538**	**0.082875458**	**0.003663**
Average		29.7	0.666699167	0.300212514	0.037375	18	0.914164174	0.07507172	0.00699005
Normalization		1	1	1	1	**0.606060606**	**1.371179416**	**0.250061928**	**0.18702456**

Take Net25 as an example, after running our approach, we totally inserted 27 buffers along edges of Net25. Also, the use on metal 1x (thinnest layer) out of total use is 91%, 8% for metal 2x, and 1% for metal 4x. However, [1] uses 59 buffers in the

whole process, and distributes 71% of total metal use on metal 1x, 27% on metal 2x, and 2% on metal 4x.

Then we take the average among the 10 trials as shown in the second to the last row in Table 2. Moreover, we obtained values of normalization by dividing our obtained average by the corresponding average values in [1]. The experimental results show that compared with [1], our approach achieved around 39% less in buffer counts use and distribute 37% more on metal 1x use, and 81% less on metal 4x. The results show that our approach save more buffer usage to reduce the power dissipation and have more flexibility to use limited thicker layer for assigning the subnets of other critical nets.

In Table 3, we set the timing constraint as $0.9\tau_{\max}$ which allows for more timing budget for interconnect optimization. We repeat the same process in Table 1 while meeting the loose timing constraint. The experimental results show that compared to [1], our approach achieved around 30.4% reduction in buffer + wire capacitance on average.

Table 3. Comparison between previous work [1] and our complete approach under timing constraint = $\mathbf{0.9}\,\tau_{max}$

Timing Constraint = 0.9 τ_{max}		LADY					Ours				
Net	Sinks	Slack(ps)	Buffer Cap(pf)	Wire Cap(pf)	Total Cap(pf)	Time(sec.)	Slack(ps)	Buffer Cap(pf)	Wire Cap(pf)	Total Cap(pf)	Time(sec.)
Net5-1	5	41.5496	0.080025	7.728571	7.808596	0.45	388.111	0.034608	5.019047	5.053655	3.4
Net5-2	5	41.5496	0.080025	7.014285	7.09431	0.45	388.111	0.034608	4.5619047	4.5965127	3.58
Net6-1	6	10380.6	0.116508	6.209523	6.326031	1.36	4925.81	0.032305	4.871429	4.903734	5.21
Net6-2	6	12758.7	0.131022	8.017142	8.148164	1.38	6575.31	0.042245	6.131428	6.173673	5.57
Net8	8	7849.72	0.114039	4.7619047	4.8759437	2.51	7341.56	0.043986	4.009523	4.053509	3.51
Net10	10	4058.17	0.133309	7.071428	7.204737	4.95	27.3453	0.02982	4.342857	4.372677	8.43
Net14	14	12805.8	0.172277	15.095238	15.267515	13.35	6679.09	0.062125	8.566666	8.628791	10.15
Net17	17	7208.05	0.213639	10.2	10.413639	23.99	4856.29	0.0497	7.109523	7.159223	14.33
Net20	20	5456.77	0.325375	13.990476	14.315851	39.49	2536.95	0.07455	10.223809	10.298359	28.28
Net25	25	8927.29	0.341812	13.304761	13.646573	77.48	173.322	0.067095	10.914285	10.98138	25.93
Average		6952.82	0.1708031	9.3393328	9.5101359	16.541	3389.19	0.0471042	6.5750471	6.6221513	10.839
Normalization		1	1	1	1	1	0.487455	0.2757807	0.7040167	**0.6963256**	0.6552808

Table 4. Comparison on buffer numbers and layer distribution between previous work [1] and our complete approach under timing constraint = $\mathbf{0.9}\,\tau_{\max}$

Timing Constraint $= 0.9\,\tau_{\max}$		LADY				Ours			
Net	Sinks	Buffer No.	Layer Distribution			Buffer No.	Layer Distribution		
			1x(%)	2x(%)	4x(%)		1x(%)	2x(%)	4x(%)
Net5-1	5	13	0.5	0.409937888	0.0900621	12	0.921325052	0.072463768	0.00621118
Net5-2	5	13	0.5393134	0.375415282	0.0852713	12	0.954595792	0.03765227	0.00775194
Net6-1	6	20	0.709219858	0.275582573	0.0151976	13	0.973657548	0.021276596	0.00506586
Net6-2	6	22	0.70734127	0.243055556	0.0496032	17	0.935515873	0.064484127	0
Net8	8	19	0.746867168	0.253132832	0	10	0.944862155	0.055137845	0
Net10	10	21	0.558652729	0.299651568	0.1416957	12	0.957026713	0.034843206	0.00813008
Net14	14	29	0.387837063	0.508892714	0.1032702	25	0.971313827	0.026965003	0.00172117
Net17	17	37	0.61167002	0.364185111	0.0241449	20	0.998658618	0.001341382	0
Net20	20	57	0.691428571	0.263333333	0.0452381	30	0.977619048	0.022380952	0
Net25	25	54	0.747252747	0.239468864	0.0132784	27	0.956959707	0.039835165	0.00320513
Average		28.5	0.619958283	0.323265572	0.0567761	17.8	0.959153433	0.037638031	0.00320854
Normalization		1	1	1	1	**0.624561404**	**1.547125767**	**0.116430683**	**0.05651203**

In Table 4, we repeat the same process in Table 2 while meeting the timing constraint $0.9\,\tau_{\max}$. The experimental results show that compared with [1], our approach achieved around 38% less in buffer counts use and distribute 55% more on metal 1x use, and 94% less on metal 4x.

In Table 5, we set the timing constraint as $30\sqrt{\tau_{\max}}$ which gives a more restrictive timing budget for each interconnect testcase. We repeat the same process in Table 1 and 3. The experimental results show that compared to [1], our approach achieved around 22.4% reduction in buffer + wire capacitance on average.

Table 5. Comparison between previous work [1] and our complete approach under timing constraint = $\gamma\sqrt{\tau_{\max}}, \gamma = 30$

Timing Constraint $= 30\sqrt{\tau_{\max}}$		LADY					Ours				
Net	Sinks	Slack(ps)	Buffer Cap(pf)	Wire Cap(pf)	Total Cap(pf)	Time(sec.)	Slack(ps)	Buffer Cap(pf)	Wire Cap(pf)	Total Cap(pf)	Time(sec.)
Net5-1	5	9.96337	0.096933	8.338095238	8.435028238	0.46	13.3173	0.068044	5.119047619	5.187091619	3.63
Net5-2	5	9.96337	0.099311	7.519047619	7.618358619	0.46	13.3173	0.065559	5.095238095	5.160797095	2.74
Net6-1	6	441.772	0.114023	8.485714286	8.599737286	1.39	272.556	0.060862	5.366666667	5.427528667	6.15
Net6-2	6	608.145	0.133416	11.27428571	11.40770171	1.41	1616.57	0.099466	7.6	7.699466	6.03
Net8	8	2702.73	0.118918	4.67619	4.795108	2.55	453.568	0.060557	5.10952381	5.17008081	8.56
Net10	10	696.726	0.14296	6.733333333	6.876293333	4.94	998.002	0.063058	5.226190476	5.289248476	64.1
Net14	14	1848.75	0.247364	12.2047619	12.4521259	13.55	1205.52	0.131161	10.34761905	10.47878005	24.96
Net17	17	1154.46	0.24753	9.442857143	9.690387143	24.26	447.202	0.097377	7.947619048	8.044996048	91.47
Net20	20	1574.98	0.303513	16.38571429	16.68922729	40.54	83.8894	0.160724	12.45238095	12.61310495	126.56
Net25	25	754.041	0.395096	12.81428571	12.81428571	77.93	287.262	0.155647	11.8952381	12.0508851	90.65
Average		980.1531	0.1899064	9.787428524	9.937825324	16.749	539.1204	0.0962455	7.615952381	7.712197881	42.485
Normalization		1	1	1	1	1	0.550037	0.506804931	0.778136194	**0.776044822**	2.53656935

Table 6. Comparison on buffer numbers and layer distribution between previous work [1] and our complete approach under timing constraint = $\gamma\sqrt{\tau_{\max}}, \gamma = 30$

Timing Constraint $= 30\sqrt{\tau_{\max}}$		LADY				Ours			
Net	Sinks	Buffer No.	Layer Distribution			Buffer No.	Layer Distribution		
			1x(%)	2x(%)	4x(%)		1x(%)	2x(%)	4x(%)
Net5-1	5	15	0.516563147	0.318840579	0.1645963	12	0.887163561	0.112836439	0
Net5-2	5	15	0.523809524	0.339977852	0.1362126	11	0.830564784	0.161683278	0.00775194
Net6-1	6	19	0.583586626	0.221884498	0.1945289	12	0.888551165	0.096251266	0.01519757
Net6-2	6	22	0.578373016	0.153769841	0.2678571	16	0.688492063	0.307539683	0.00396825
Net8	8	20	0.766917293	0.228070175	0.0050125	9	0.72556391	0.239348371	0.03508772
Net10	10	22	0.594657375	0.286875726	0.1184669	10	0.799651568	0.163182346	0.03716609
Net14	14	40	0.687894435	0.232931727	0.0791738	23	0.780837636	0.205393001	0.01376936
Net17	17	42	0.718309859	0.257545272	0.0241449	19	0.883299799	0.11535882	0.00134138
Net20	20	53	0.591904762	0.292857143	0.1152381	32	0.79	0.192380952	0.01761905
Net25	25	62	0.77518315	0.221153846	0.003663	29	0.867216117	0.127289377	0.00549451
Average		31	0.633719919	0.248340676	0.1108894	17.3	0.81413406	0.172126353	0.01373959
Normalization		1	1	1	1	**0.558064516**	**1.28469066**	**0.693105762**	**0.1239035**

In Table 6, we repeat the same process in Table 2 and 4 while meeting the timing constraint $30\sqrt{\tau_{\max}}$. The experimental results show that compared with [1], our approach achieved around 44% less in buffer counts use and distribute 28% more on metal 1x use, and 88% less on metal 4x.

5. CONCLUSION

In this thesis, we proposed a Lagrangian relaxation method to minimize the usage of buffers and metal resources while satisfying the timing constraints. As such we are able to reduce the total power consumption and save the limited thicker layer for assigning the subnets of other critical nets with futher timing improvement.

Compared with the previous work [1] that used traditional Van Ginneken's algorithm that allows for bumping up of thin to thick wire given significant delay improvment, our approach achieved around 25% reduction in buffer + wire capacitance.

REFERENCES

[1] Z. Li, C. Alpert, S. Hu, T. Muhmud, S. Quay, and P. Villarrubia. "Fast interconnect synthesis with layer assignment," *in Proc. of ISPD*, pp. 71-77, 2008.

[2] C. C. N. Chu and D. F. Wong. "A new approach to simultaneous buffer insertion and wire sizing," *in Proc. of ICCAD*, pp. 614-621, 1997.

[3] J. Lillis, C. K. Cheng, and T. T. Y. Lin. "Optimal wire sizing and buffer insertion for low power and a generalized delay model," *in Proc. of ICCAD*, pp. 138-143, 1995.

[4] I. M. Liu, A. Aziz, and D. F. Wong. "Meeting delay constraints in DSM by minimal repeater insertion," *in Proc. of DATE*, pp. 436-440, 2000.

[5] C.J. Alpert and A. Devgan. "Wire segmenting for improved buffer insertion," *in Proc.of DAC*, pp. 588-593, 1997.

[6] C. J. Alpert, M. Hrkic, and S. T. Quay. "A fast algorithm for identifying good buffer insertion candidate locations," *in Proc. of ISPD*, pp. 47-52, 2004.

[7] L. P. P. P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal Elmore delay," *in Proc. of IEEE ISCAS*, pp. 865-868, 1990.

[8] A.B. Kahg, S. Muddu, E. Sarto, and R. Sharma. "Interconnect tuning strategies for high-performance ICs," *in Proc. of DATE*, pp. 471-478, 1998.

[9] Y. Peng and X. Liu. “Freeze: Engineering a fast repeater insertion solver for power minimization using the ellipsoid method,” *in Proc. of DAC*, pp. 813-818, 2005.

[10] S. Hu, Z. Li, and C. J. Alpert. “A polynomial time approximation scheme for timing constrained minimum cost layer assignment,” *in Proc. of ICCAD*, pp.112-115, 2008.

[11] G. S. Garcea, N. P. van der Meijs, and R. H. Otten. “Simultaneous analytical area and power optimization for repeater insertion,” *in Proc. of ICCAD*, pp. 568-573, 2003.

[12] C. Chen, C. C. N. Chu, and D. F. Wong. “Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation,” *IEEE Trans. CAD*, vol. 18, no. 7, pp. 1014-1025, 1999.

[13] M. Ozdal, S. Burns and J. Hu. “Gate sizing and device technology selection algorithms for high-performance industrial designs,” *in Proc.of ICCAD*, pp. 724-731, 2011.

[14] Y. L. Huang, J. Hu and W. Shi. “Lagrangian relaxation for gate implementation selection,” *in Proc. of ISPD*, pp. 167-174, 2011.

[15] Y. J. Ho and W. K. Mak. “Power and density-aware buffer insertion,” *in Proc. of VLSI-DAT*, pp. 287-290, 2008.

[16] S. Hu, C. J. Alpert, J. Hu, S. K. Karandikar, Z. Li, W. Shi, and C. N. Sze. “Fast algorithms for slew-constrained minimum cost buffering,” *IEEE Trans. CAD*, vol. 26, no. 11, pp. 2009-2022, 2007.