UTACO: A Unified Timing and Congestion Optimizing Algorithm for Standard Cell Global Routing*

Tong Jing, Xianlong Hong, Haiyun Bao,
Yici Cai, Jingyu Xu,

Dept. of CST
Tsinghua Univ.
Beijing 100084, P. R. China
Tel: +86-10-62785564
Fax: +86-10-62781489
e-mail: {jingtong, hxl-dcs}@tsinghua.edu.cn

Chungkuan Cheng

Dept. of CSE
UC, San Diego
La Jolla, CA 92093-0114, USA
Tel: +1-858-534-6184
Fax: +1-858-534-7029
e-mail: kuan@cs.ucsd.edu

Jun Gu

Dept. of CS
HK Univ. of S&T
Hong Kong, P. R. China
Tel: +86-10-82624605
Fax: +86-10-82624604
e-mail: gu@cs.ust.hk

Abstract -- Timing performance and routability are two main issues of global routing. In this paper, we adopt a shadow price mechanism to incorporate the two issues into one unified objective function. The shadow price of a net is the sum of its congestion price and timing price. Based on the new formulation, this paper presents the UTACO algorithm for standard cell (SC) global routing. The experimental results show that UTACO is efficient for both timing and congestion optimization.

I. INTRODUCTION

As fabrication technology moves into deep sub-micron (DSM) device size and giga-hertz clock frequencies, interconnect becomes an increasingly dominant factor in performance, power, reliability and cost [1]. Especially, interconnect has become a dominant factor in chip timing [2]. Thus, merely minimizing congestion throughout global routing process is not adequate [3]. We need efficient timing and congestion optimizing algorithms for global routing.

To deal with this trend, many helpful researches, such as timing models [4, 5], timing-driven Steiner tree algorithms [6, 7], and interconnect design algorithms considering buffer insertion or/and wire sizing [8-10], have been done to reduce interconnect delay. These researches mainly focus on single net routing. To get better routing results for timing optimization, we also need efficient timing-driven global routing algorithms. Some typical timing-driven global routing algorithms [3, 11-16] have been presented.

Ref. [11] proposes the nets-based timing analysis strategy to improve the timing performance. This strategy has advantages compared with previous methods: (1) simply transform the minimum interconnect delay into the minimum wire length [12, 13] and (2) assign higher priorities to more critical nets by static delay analysis and then obtain shorter routes for each net [14, 15]. This approach is easy to implement and simple to control. However, this strategy has blindness in delay assignment. If there is a difference between delay assignment and timing requirement, the congested nets can not be rerouted.

To overcome the shortcomings in [11], Ref. [16] presents the critical-path-based timing analysis strategy. Throughout the global routing process, many nets can be rerouted to slack the congested areas while the timing constraints on critical paths are still satisfied. This method avoids the unreasonable delay assignment and obtains a better routing result. Since there are many critical paths actually, this strategy takes much time to check all critical paths after each net is rerouted. Thus, it reduces routing speed greatly. Meanwhile, net ripping up and rerouting only depends on greedy trying. There is not an overall survey to reduce the delay. The reason is that there is not a unified formulation for both timing and congestion optimization. That is, the formulation only consists of equations for congestion minimizing. Therefore, the whole optimizing result is limited.

In later research [3], both timing and congestion can be optimized. This method does not adopt a unified formulation for both timing and congestion optimization. It is a timing-constrained global routing algorithm for BBL (building block layout) instead of SC (standard cell). The testing benchmarks are in small scale and the algorithm has a longer run time, which limits the algorithm applications.

Timing performance and routability are two main issues of global routing. However, these two issues are mutually conflicting if we view and handle their effects independently. For example, to ease the routing congestion of a local area, we tend to detour wires out of the area for a more even distribution of the routing demand. However, the detour of the wires lengthens the interconnect distances and increases the signal delay. Likewise, timing driven routing would enforce certain routes to optimize the communication speed, which might cause local routing congestion.

To tackle this problem and get good routing results, this paper presents a unified timing and congestion optimizing (UTACO) algorithm for SC global routing. The UTACO algorithm adopts a shadow price mechanism (see [17] for an overview) to incorporate timing and congestion optimizing into one unified objective function. The shadow price of a net is the sum of its congestion price and timing price. The timing analysis strategy in UTACO is different from that in the above mentioned approaches. The performance of UTACO is better than that of the existing algorithms for timing optimization, which makes it possible: (1) to optimize timing and congestion simultaneously and efficiently, (2) to reduce the delay in an overall survey.

The remainder of this paper is organized as follows. In Section 2, the Sakurai-delay-based timing model for

* This work was supported partly by Hi-Tech Research and Development (863) Program of China 2002AA1Z1460, the NSF MIP-9987678, the 973 Program of China G1998030403, the NSFC 60121120706, the NSF CCR-0096383, and Key Faculty Support Program of Tsinghua Univ. [2002] 4.

calculating the delay is introduced. Section 3 formulates the global routing problem for symbolic analysis. The UTACO algorithm is given in detail in Section 4. Section 5 discusses experimental results. Section 6 is an overall conclusion.

II. Timing Model

Timing optimizing global routing needs a suitable timing model to calculate the delay in a routing tree. The Sakurai-delay-based timing model [5] is used in this paper.

III. Problem Formulation

Ref. [18-20] introduce the multi-commodity flow into global routing. The goal of these algorithms is only to minimize congestion. Timing optimization does not be considered and formulated. In this section, based on shadow price mechanism, we formulate global routing as a multi-commodity flow problem and incorporate timing and congestion optimizing into one unified objective function.

In this paper, the objective function is the slack of congestion with the clock period as the delay limit that from registers and inputs to registers and outputs. The multi-commodity flow is expressed by a linear programming (see [17] for an overview) formulation as a primal problem. We then convert the primal problem into a dual formulation using the shadow price as the variables. In the dual formulation, the wiring congestion is reflected by the congestion price at each edge of the global routing graph. The congestion price of a net is defined to be the sum of the congestion prices on the edges passed by that net. The signal delay is reflected by the timing price at each net. If we view each timing price as a timing flow on each net, the timing flow forms paths flowing from inputs and registers to registers and outputs. The amount of the timing flow on each path corresponds to the criticality of the timing constraint on that path.

The shadow price of a net is the sum of its congestion price and timing price. The objective of the dual problem is to maximize the sum of shadow prices of all nets together with the clock period limit on the boundary of the circuit.

The primal and dual formulation offers theoretical upper and lower bounds of the routing solution. Throughout the optimization process, the difference of the two bounds reduces. When the difference approaches zero, we have an optimal solution. However, the amount of routing flow is limited by discrete numbers, the difference always exists. The bounds thus provide the user's insight into the quality of the solutions.

A. Definitions

With the progress in multi-layer routing technology, routing area is a whole chip plane instead of many channels. We assume that the whole chip is divided into a rectangular array of N_{row}*N_{col} cells called global routing cells (GRCs). Global routing graph (GRG) is the dual graph of GRCs, which is composed of the gridlines and crossings. Fig.1 shows an example GRG that holds 4*4 GRCs. Node v_i represents the center point of GRC$_i$. The edge links node v_i and node v_j is named as e, l is called the length of edge e, equals the distance between node v_i and node v_j. A non-negative number c_e, called edge capacity, is assigned to edge e. c_e indicates the number of available tracks between every two corresponding GRCs.

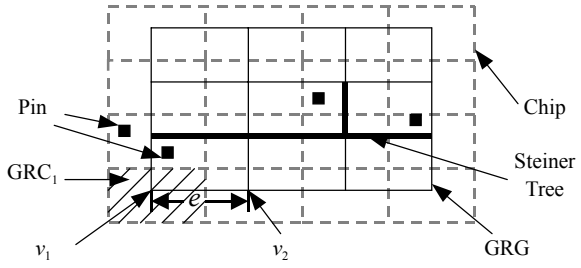


Fig.1. Global routing graph (GRG).

Thus, a net can be specified as a set of nodes in GRG. Then, the problem of routing a net in GRG can be described as a Steiner tree problem of specified nodes in GRG.

B. Notation

The following notation is used in this paper.

N	the set of nets to be completed, assume that these nets are numbered consecutively, i.e., 1 through r.
n	a set of terminals to be routed under the timing constraint, each net $n \in N$.
E	edges in GRG, each edge $e \in E$.
t_n^f	the flow f of net n.
b_n	the demand of routing for net n, in general, is set to 1.
n_s	the source of net n.
n_t	a sink of net n.
d_{ij}^f	the delay of flow t_n^f based on the Sakurai-delay-based timing model ($\forall\, i \in n_s, j \in n_t$).
a_i	the latest arrival time of node i.
P	the timing constraint.
$\varphi_e(t_n^f)$	a function is used to indicate whether the flow t_n^f goes through edge e or not. We have $\varphi_e(t_n^f) = 1$ if the flow t_n^f goes through edge e; otherwise, $\varphi_e(t_n^f) = 0$.
r_e	the shadow cost, i.e., the weight value of edge e.
λ_n	the shadow cost of routability flow of net n.
ω_{ij}	the shadow cost of timing flow of net n ($\forall\, i \in n_s, j \in n_t$).
u_i	the shadow cost of node i.
$\lambda_{t_n^f}$	the maximum cost of flow t_n^f with respect to routability and timing requirements.

C. Linear Programming Formulation

We use a factor s to scale the edge capacity c_e of the edges to $s \bullet c_e$. The flow problem searches for a flow solution that minimize slack s. Thus, if the solution has $s \le 1$, we can route the nets with a slack $(1\text{-}s) \bullet c_e$. Otherwise, when solution has $s>1$, the routing is not feasible.

Minimize s
Subject to

$$r_e: \quad -\sum_{f,n\in N} t_n^f \cdot \varphi_e(t_n^f) + s \cdot c_e \ge 0 \qquad \forall\, e \in E \qquad \text{(A)}$$

$$\lambda_n: \quad \sum_f t_n^f \ge b_n \qquad \forall\, n \in N \qquad \text{(B)}$$

ω_{ij}: $-a_i - \sum_f \frac{1}{b_n} \cdot d_{ij}^f \cdot t_n^f + a_j \geq 0$

$\forall\, i \in n_s, j \in n_t, \forall\, n \in N$ (C)

u_i: $-a_i \geq -P$ $\forall$ node i (D)

$a_i \geq 0,\ t_n^f \geq 0,\ s \geq 0$ $\forall$ node $i,\ n \in N$

The constraint (A) of the linear program specifies that the total demands of the nets using the edge e should be less than the edge capacity c_e. The constraint (B) specifies that the sum of all of the routing tree flows must be more than or equal to the demand for that net. The constraint (C) specifies the edge timing constraint, and the constraint (D) specifies the pin timing constraint.

D. Dual Linear Programming Formulation

For the linear programming, we can derive its dual formulation as follows.

Maximize $(\sum_{n\in N} \lambda_n \cdot b_n - \sum_i u_i \cdot P)$ (E)

Subject to

s: $\sum_{e\in E} c_e \cdot r_e \leq 1$ (F)

t_n^f: $-\sum_{e\in E} r_e \cdot \varphi_e(t_n^f) + \lambda_n - \frac{1}{b_n} \cdot \sum_{i\in n_s, j\in n_t} d_{ij}^f \cdot \omega_{ij} \leq 0$

$\forall\, n \in N$ (G)

a_i: $-\sum_j \omega_{ij} + \sum_j \omega_{ji} - u_i \leq 0$ $\forall$ node i (H)

$\lambda_n \geq 0,\ \omega_{ij} \geq 0$ $\forall\, n \in N, i \in n_s, j \in n_t$

$u_i \geq 0,\ r_e \geq 0$ $\forall$ node i, $e \in E$

The constraint (F) is derived from the primal problem with respect to the variable s, the constraint (G) is derived with respect to the variable t_n^f, and the constraint (H) is derived with respect to the a_i.

IV. The Global Routing Algorithm UTACO

In global routing, we require that the flow amount is discrete, i.e., flow t_n^f is in the set of natural integers. However, this transforms the problem into an integer program, which is known to be NP-complete. Thus, we firstly focus on obtaining a fractional solution. For further discussion about heuristic integerizing, refer to Ref. [18].

A. Fractional Flow Algorithm

(1) Given both shadow cost of edge e and shadow cost of timing flow of net n, the primal operation reroutes the nets to optimize congestion, find the optimal timing Steiner trees, and obtain the flow $\sum_{f,n\in N} t_n^f \cdot \varphi_e(t_n^f)$. The weight value of edge e is regarded as the shadow cost function (i.e., r_e) of it, and the shadow cost of timing flow of net n is ω_{ij} ($\forall\, i \in n_s, j \in n_t$).

The dual operation is used to calculate edge cost r_e according to edge congestion and to calculate net cost ω_{ij} under the timing constraint (H). After the primal iteration, we have the flow amount of each edge, i.e., $\sum_{f,n\in N} t_n^f \cdot \varphi_e(t_n^f)$, and the delay information of each critical path. The flow on each edge is used to define the congestion. And, based on the information, we can manage to get the suitable set of edges to be rerouted.

The flow control of the fractional flow algorithm can be summarized as follows: we firstly set an initial routing by routing each net independently, and then iterate the primal dual process until the two objective functions converge or the number of iterations exceeds the limit.

(2) From equation (G) and (H), we have:

t_n^f: $\lambda_n \leq \sum_{e\in E} r_e \cdot \varphi_e(t_n^f) + \frac{1}{b_n} \cdot \sum_{i\in n_s, j\in n_t} d_{ij}^f \cdot \omega_{ij}$

$\forall\, n \in N$ (I)

a_i: $\sum_j \omega_{ji} \leq \sum_j \omega_{ij} + u_i$ $\forall$ node i (J)

The objective from (E) in dual linear programming prescribes that the net cost λ_n needs to reach its upper bound. Thus, we can use "=" instead of "≤" in equation (I). On the other hand, it should be satisfied equation (I) for all flows t_n^f. It means that we need the minimum value of the right part of equation (I) among all its values. Thus, from (I), we have:

$\lambda_n = \min_{f,n}(\sum_{e\in E} r_e \cdot \varphi_e(t_n^f) + \frac{1}{b_n} \cdot \sum_{i\in n_s j\in n_t} d_{ij}^f \cdot \omega_{ij})$

$\forall\, n \in N$ (K)

Let $\lambda_{t_n^f} = (\sum_{e\in E} r_e \cdot \varphi_e(t_n^f) + \frac{1}{b_n} \cdot \sum_{i\in n_s j\in n_t} d_{ij}^f \cdot \omega_{ij})$

$\forall\, n \in N$ (L)

Then, $\lambda_n = \min_{f,n}(\lambda_{t_n^f})$ $\forall\, n \in N$ (M)

We can regard $\lambda_{t_n^f}$ in equation (L) as the cost of flow t_n^f. Thus, net cost λ_n is the minimum cost of all possible flows (see equation (M)) in net n. Flow cost $\lambda_{t_n^f}$ depends on two parts: the value of $\sum_{e\in E} r_e \cdot \varphi_e(t_n^f)$ and $\frac{1}{b_n} \cdot \sum_{i\in n_s j\in n_t} d_{ij}^f \cdot \omega_{ij}$. Of the two parts, the former is the *congestion cost*, and the latter is the *delay cost*.

For dual programming, in order to maximize the congestion cost $\sum_{e\in E} r_e \cdot \varphi_e(t_n^f)$, the edge cost r_e is set higher for the more congested edge. Likewise, to maximize the delay cost, the shadow cost ω_{ij} is set higher for the longer delay d_{ij}^f.

On the other hand, the primal problem then routes the flow to minimize the congestion cost and the delay cost. Thus, the nets through edges with high shadow cost r_e will be rerouted to find the path along less congested regions. In the mean time, the flows along the high shadow cost ω_{ij} will be rerouted to reduce the delay. The shadow prices derived from the dual problem provide the mechanism to balance the effects between the routing congestion and the timing delay

optimization.

(3) To get the maximum net cost, shadow cost ω_{ij} has to reach its upper bound (in equation (K)). Thus, the inequality in expression (J) can be replaced by the equality:

$$a_i: \quad \sum_j \omega_{ji} = \sum_j \omega_{ij} + u_i \qquad \forall \text{ node } i \qquad \text{(N)}$$

If we regard net cost ω_{ij} as the flow from node i to node j, it means that the input flow equals the output flow of node i, shown in Fig. 2.

From equation (D), we know that if $a_i > P$, then, $u_i > 0$. Otherwise, $a_i \le P$, then, $u_i = 0$.

According to equation (C), if edge e is not on the critical path of timing, $\omega_{ij} = 0$. Otherwise, $\omega_{ij} > 0$.

An illustration of net cost ω_{ij} is shown in Fig. 3. Fig. 3(a) shows the latest arrival time of each node. We can see that a_i equals 5. Suppose that the timing constraint P equals 4. We have the critical path shown by the bold arrow lines. Fig. 3(b) shows the relationship between net cost ω_{ij} and node cost u_i. In Fig. 3(b), let u_i of the node that violates the timing constraint be 2. The total flow ω_{ij} is 2. For the edge e that is not on the critical path, $\omega_{ij} = 0$.

(4) From section 4A**(2)**, we know that we need to get the maximum value of net cost λ_n for the dual linear programming. But, in the primal linear programming, it needs to reroute net n with minimum shadow cost λ_n. Therefore, by using the method of iterating the primal dual process, we will choose the nets with $\Delta_{\max}\lambda = \max_{f,n}(\lambda_{t_n^f}^{(k)} - \lambda_{t_n^f}^{(k+1)})$ to reroute.

In order to choose the nets with $\Delta_{\max}\lambda$ to reroute, we can derive the following equation (O) for $\Delta_{\max}\lambda$ according to equation (K) and (L).

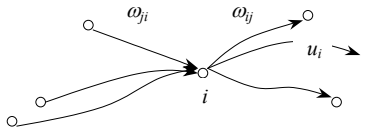
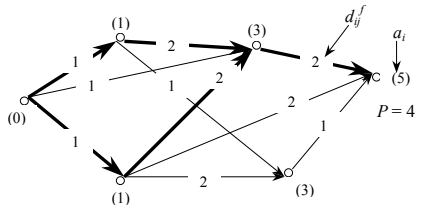
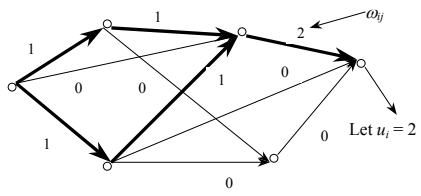


Fig.2. The flow of node i.



(a) a_i and critical path



(b) ω_{ij} and u_i

Fig.3. An illustration of net cost ω_{ij}.

$$\Delta_{\max}\lambda = \max_{f,n}(\lambda_{t_n^f}^{(k)} - \lambda_{t_n^f}^{(k+1)})$$

$$= \max_{f,n}((\sum_{e\in E} r_e^{(k)} \cdot \varphi_e(t_n^f) + \frac{1}{b_n} \cdot \sum_{i\in n_s, j\in n_t} d_{ij}^f \cdot \omega_{ij}^{(k)}) -$$

$$(\sum_{e\in E} r_e^{(k+1)} \cdot \varphi_e(t_n^f) + \frac{1}{b_n} \cdot \sum_{i\in n_s, j\in n_t} d_{ij}^f \cdot \omega_{ij}^{(k+1)}))$$

$$\forall\, n \in N \quad \text{(O)}$$

Based on equation (O), the rerouting is helpful for both congestion and timing optimization. Separately, we might use $\max(r_e^{(k)} - r_e^{(k+1)})$ and $\max(\omega_{ij}^{(k)} - \omega_{ij}^{(k+1)})$. A uniform routing result, i.e., a low routing density, will make edge cost r_e be decreased. Then, we will get $\max(r_e^{(k)} - r_e^{(k+1)})$. On the other hand, if the rerouted edge e is on the critical path of timing, then, $\omega_{ij} > 0$. And, the rerouted result for edge e will make the path is not "critical" than before. Accordingly, net cost ω_{ij} is decreased. Thus, we will get $\max(\omega_{ij}^{(k)} - \omega_{ij}^{(k+1)})$. However, equation (O) shows the integration of the routability and timing requirements into one unified expression.

(5) As a result, we have that the new formulation obtains a best result for both congestion minimization and timing optimization. In the following, we will briefly discuss the suitable method of getting the nets with $\Delta_{\max}\lambda$ to reroute.

B. The CC-net

In section 4A**(4)**, it comes to the conclusion that we will manage to get the nets with $\Delta_{\max}\lambda$ to reroute. In this subsection, we will give a method to select these nets.

Based on the information about congested edges and critical paths given by the primal iteration, we create a network *NW*. *NW*= (V_{cc}, E_{cc}, $\Delta_{\max}\lambda$, PI, PO), which consists of and only consists of all congested edges and critical paths. The network *NW* is called CC-net. Where V_{cc} is the set of pins of *NW*, E_{cc} is the set of edges of *NW*, PI is the source of *NW* and PO is the sink of *NW*. Thus, if we reroute the edges in *NW*, both congestion and timing can be optimized to a certain extent.

Fig. 4 shows the partial CC-net in MCNC C2, where there are total 16 CC-paths from PI to PO. In MCNC C7, there are total more than 45 thousand such CC-paths throughout routing process. Under the critical-path-based timing optimizing approach [16], since there is not any unified formulation and it is in order to shorten run time, only some critical paths are checked with timing constraints. Therefore, the optimizing result is limited. Now, we create the CC-net. We can get $\Delta_{\max}\lambda$ without checking every CC-path in *NW*, even without checking each critical path. How can we get $\Delta_{\max}\lambda$ in CC-net? The answer is in the followings.

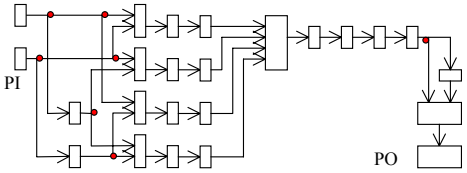


Fig.4. Partial CC-net in MCNC C2.

From all the CC-paths from PI to PO in the CC-net, if we select at least one edge from each CC-path, reduce the delay and minimize the congestion of these selected edges, then we certainly improve the timing performance of all critical paths and minimize congestion. Thus, we get $\Delta_{\max}\lambda$. That is, we should select the min-cut of the CC-net. Then, decrease the delay and congestion of all edges in the min-cut. As a result, we are able to get the nets with $\Delta_{\max}\lambda$.

Based on the theory of maximum network flow [22], we can get the maximum flow of the CC-net. If the value of the maximum flow does not equal $\propto$, we will get its min-cut.

C. Edge Shadow Cost

In this subsection, we formulate the edge shadow cost r_e. The cost r_e describes the routing congestion. We formulate r_e as follows.

$$C_g = (\sum_{f,n} t_n^f \cdot \varphi_e(t_n^f) + \eta\delta)/(c_e + \delta) \qquad \text{(P)}$$

$$r_e = \begin{cases} C_g & \sum_{f,n} t_n^f \cdot \varphi_e(t_n^f) \le c_e \\ \mathrm{K} \cdot C_g & otherwise \end{cases} \qquad \text{(Q)}$$

where η is an adjuster ($0 < \eta < 1$), K is a large integer, and δ is a small real number that validates the equation (P) while edge capacity c_e is 0.

D. The UTACO Algorithm

Based on the above discussion, the essence of the UTACO algorithm can be summarized in Fig. 5.

ALGORITHM UTACO
S1: construct initial minimum wire length Steiner Tree for each net respectively;
S2: make statistics of total resources, determine congested edges and critical paths;
WHILE (solution is not ε-optimal) **DO**
{
S3 S31: calculate the edge shadow cost r_e;
S32: get the value of net cost ω_{ij} for each edge;
S33: get the nets with $\Delta_{\max}\lambda$;
S4 S41: reroute the nets given by step S33 to implement congestion and timing optimizing;
S42: update the flow demand of each edge;
S5: check if solution is ε-optimal;
}
S6: solution integerization;
WHILE (not feasible) **DO**
{
S7: improve integerized solution;
}

Fig.5. The UTACO algorithm.

V. Experimental Results and Discussions

The UTACO algorithm has been implemented in the C language on a Sun Ultra Enterprise 450. We compare routing results between UTACO and SSTT (search space traversing technology) algorithm on main optimization objectives. The SSTT algorithm presented in [21] is a pure congestion minimization algorithm for SC global routing. Ref. [21] shows that SSTT can get better routing results on wire length and overflow edges. SSTT also has a very short run time. The experimental results are also compared with those of above algorithms.

A. Benchmark Data

We tested five MCNC benchmarks under 0.2*um* technology. Table 1 summarizes the benchmark data sets.

TABLE I
BENCHMARK DATA

Circuits	Nets	Grids	Allowable Delay (*ns*)
C2	745	9*11	0.496281
C5	1764	16*18	1.045857
C7	2356	16*18	1.083933
s13207	4953	24*26	1.022260
avq	21851	65*67	4.023312

B. Results

TABLE II
CIRCUIT DELAY INCREMENTS (*ns*)

Circuits	SSTT		UTACO	
C2	0.0155	(+3.12%)	0.0000	(0.00%)
C5	0.0803	(+7.68%)	-0.0033	(-0.32%)
C7	0.0300	(+2.77%)	-0.0045	(-0.42%)
s13207	0.0711	(+6.96%)	-0.0018	(-0.18%)
avq	0.0345	(+0.86%)	-0.0142	(-0.35%)

TABLE III
THE TOTAL WIRE LENGTH (*um*)

Circuits	SSTT	UTACO	
C2	$5.2215*10^5$	$5.2194*10^5$	(-0.04%)
C5	$1.1489*10^6$	$1.1466*10^6$	(-0.20%)
C7	$1.3494*10^6$	$1.3510*10^6$	(+0.12%)
s13207	$7.8266*10^6$	$7.8179*10^6$	(-0.11%)
avq	$8.2860*10^6$	$8.2818*10^6$	(-0.05%)

TABLE IV
OVERFLOW EDGES

Circuits	SSTT	UTACO
C2	1.7	2.2
C5	1.3	1.6
C7	0.0	0.6
s13207	3.8	6.8
avq	13.8	10.0

TABLE V
THE RUNNING TIME (*s*)

Circuits	SSTT	UTACO
C2	1.28	1.78
C5	3.05	4.72
C7	4.59	7.04
s13207	21.60	28.13
avq	50.79	70.36

C. Discussions

(1) The main optimizing objective of UTACO is to reduce circuit delay, which is the essential difference between timing-optimizing and non-timing-optimizing global routing algorithms. As a result, the increment (i.e., the slack value) between actual delay and allowable delay (i.e., actual delay minus allowable delay) is given in table 2. Table 2 also gives the percentage of delay increments. Table 2 indicates that the delay in routing results is fluctuant and may be greater than the allowable value while it does not be controlled in SSTT. When

UTACO algorithm is used, the circuit delay can be controlled efficiently within the range of the required value.

Ref. [11] and [16] test the MCNC C2, C7, and s13207 under 2*um* technology. Their routing results show that the maximum delay is better improved than that of TimberWolf 5.6. However, they do not compare their results with the allowable delay value. Thus, designers do not know whether the delay can satisfy delay requirements or not by using those algorithms.

(2) One of the optimizing objectives of UTACO is to shorten wire length. The wire length results are given in table 3. The last column in table 3 gives the percentage of wire length increments between UTACO and SSTT. We find that there is only a little difference of wire length between UTACO and SSTT, which indicates that UTACO performs well on timing optimizing and does not make wire length get worse. There are no experimental results in [11, 16] concerning this test.

(3) The UTACO algorithm also includes the objective of removing overflow edges. In table 4, we find that the number of overflow edges in UTACO nearly equals to that in SSTT, which indicates that UTACO performs well on both timing and congestion (shown by overflow edges) optimizing. There are no experimental results in [3, 11, 16] concerning this test.

(4) The running time is given in table 5. It shows that UTACO takes a very short running time, which is from 2s to 71s. Thus, the UTACO algorithm can be used for the large scale circuit global routing. The reason is that we adopt the unified timing and congestion optimizing strategies to improve the circuit timing performance and routability. The existing algorithms [3, 11, 16] take much longer run time (approximate 200s ~ 3000s) or are tested on very small scale benchmarks (45 ~ 390 nets).

VI. Conclusion and Future Work

This paper studies the timing performance and routability in global routing. A unified timing and congestion optimizing (UTACO) algorithm for SC global routing is proposed. The experimental results show that the UTACO algorithm is able to: (1) optimize both timing and congestion simultaneously and efficiently, (2) reduce the delay in an overall survey, (3) obtain good routing results on other optimizing objectives, such as wire length, overflow edges, (4) take a very short running time.

Coupling and crosstalk are new challenges to VLSI/ULSI and system-on-a-chip (SOC) routing. We have not focused on coupling and crosstalk throughout the global routing process in this paper. Thus, as future work we plan to study coupling and crosstalk in detail. Then, we will integrate the functions considering coupling and crosstalk effects into our UTACO global router.

Acknowledgements

This paper describes research work done cooperatively at Tsinghua University, Beijing, P. R. China and University of California, San Diego (UCSD), USA. The authors wish to thank Hongyu Chen, Bo Yao, Zhengyong Zhu, Zhanhai Qin and Bao Liu in UCSD for valuable discussions.

References

[1] R. Kastner, E. Bozorgzadeh, M. Sarrafzadeh, "An Exact Algorithm for Coupling-Free Routing", *In: Proceedings of ACM ISPD*, Sonoma, CA, pp.10-15, 2001.

[2] T. Jing, X. L. Hong, Y. C. Cai, H. Y. Bao, J. Y. Xu, "The Key Technologies and Related Research Work of Performance-Driven Global Routing", *J. of Software*, 12(5), pp.677-688, 2001.

[3] J. Hu, S. S. Sapatnekar, "A Timing-constrained Algorithm for Simultaneous Global Routing of Multiple Nets", *In: Proceedings of IEEE/ACM ICCAD*, San Jose, CA, pp.99-103, 2000.

[4] W. C. Elmore, "The Transient Response of Lumped Linear Networks with Particular Regard to Wideband Amplifiers", *Journal of Applied Physics*, 19(1), pp.55-59, 1948.

[5] T. Sacurai, "Approximation of Wiring Delay in MOSFET LSI", *IEEE Journal of Solid-State Circuits*, 18(4), pp.418-426, 1983.

[6] X. L. Hong, T. X. Xue, E. S. Kuh, C. K. Cheng, J. Huang, "Performance-Driven Steiner Tree Algorithm for Global Routing", *In: Proceedings of ACM/IEEE DAC*, Dallas, Texas, pp.177-181, 1993.

[7] J. Y. Xu, X. L. Hong, T. Jing, Y. C. Cai, J. Gu, "An Efficient Hierarchical Timing-Driven Steiner Tree Algorithm for Global Routing", *In: Proceedings of IEEE/ACM ASP-DAC*, Bangalore, India, pp.473-478, 2002.

[8] J. Cong, L. He, K. Y. Khoo, C. K. Koh, Z. G. Pan, "Interconnect Design for Deep Submicron ICs", *In: Proceedings of IEEE/ACM ICCAD*, San Jose, CA, pp.478-485, 1997.

[9] C. C. N. Chu, D. F. Wong, "An Efficient and Optimal Algorithm for Simultaneous Buffer and Wire Sizing", *IEEE Trans. on CAD*, 18(9), pp.1297-1304, 1999.

[10] J. Lillis, C. K. Cheng, "Timing Optimization for Multisource Nets: Characterization and Optimal Repeater Insertion", *IEEE Trans. on CAD*, 18(3), pp.322-331, 1999.

[11] J. Huang, X. L. Hong, C. K. Cheng, E. S. Kuh, "An Efficient Timing-Driven Global Routing Algorithm", *In: Proceedings of ACM/IEEE DAC*, Dallas, Texas, pp.596-600, 1993.

[12] Y. Fujihara, Y. Sekiyama, Y. Ishibashi, M. Yanaka, "DYNAJUST: An Efficient Automation Routing Technique Optimizing Delay Conditions", *In: Proceedings of ACM/IEEE DAC*, Las Vegas, Nevada, pp.791-794, 1989.

[13] M. A. B. Jackson, E. S. Kuh, M. Marek-Sadowska, "Timing Driven Routing for Building Block Layout", *In: Proceedings of IEEE ISCAS*, pp.518-519, 1987.

[14] A. E. Dunlop, V. D. Agrawal, D. N. Deutsch, M. F. Jukl, P. Kozak *et al*, "Chip Layout Optimization Using Critical Path Weighting", *In: Proceedings of ACM/IEEE DAC*, Albuquerque, New Mexico, pp.133-136, 1984.

[15] M. Rose, M. Wiesel, D. Kirkpatrick, N. Nettleton, "Dense, Performance Directed, Auto Place and Route", *In: Proceedings of IEEE CICC*, Rochester, NY, pp.11.1.1-11.1.4, 1988.

[16] X. L. Hong, T. X. Xue, J. Huang, C. K. Cheng, E. S. kuh, "TIGER: An Efficient Timing-Driven Global Router for Gate Array and Standard Cell Layout Design", *IEEE Trans. on CAD*, 16(11), pp.1323-1330, 1997.

[17] D. G. Luenberger, *Linear and Nonlinear Programming*, Second Edition, Addison Wesley, 1984.

[18] R. C. Carden IV, J. M. Li, C. K. Cheng, "A Global Router with a Theoretical Bound on the Optimal Solution", *IEEE Trans. on CAD*, 15(2), pp.208-216, 1996.

[19] C. Albrecht, "Provably Good Global Routing by a New Approximation Algorithm for Multicommodity Flow", *In: Proceedings of ACM ISPD*, San Diego, CA, pp.19-25, 2000.

[20] E. Shragowitz, S. Keel, "A global router based on a multi-commodity flow model", *Integration, the VLSI J.*, 5, pp.3-16, 1987.

[21] T. Jing, X. L. Hong, H. Y. Bao, Y. C. Cai, J. Y. Xu *et al*, "An Efficient Congestion Optimization Algorithm for Global Routing Based on Search Space Traversing Technology", *In: Proceedings of IEEE International Conference on ASIC*, Shanghai, China, pp.114-117, 2001.

[22] R. E. Tarjan, "Algorithms for Maximum Network Flow", *Mathematical Programming Study*, 26, pp.1-11, 1986.