

UST/DME: A Clock Tree Router For General Skew Constraints

CHUNG-WEN ALBERT TSAO
Celestry Design Technologies, Inc.
and
CHENG-KOK KOH
Purdue University

In this article, we propose new approaches for solving the useful-skew tree (UST) routing problem [Xi and Dai 1997]: clock routing subject to general skew constraints. The clock layout synthesis engine of our UST algorithms is based on the deferred-merge embedding (DME) paradigm for the zero-skew tree (ZST) [Edahiro 1992; Chao et al. 1992] and bounded-skew tree (BST) [Cong and Koh 1995; Huang et al. 1995; Kahng and Tsao 1997; Cong et al. 1998] routings; hence, the names UST/DME and Greedy-UST/DME for our UST algorithms. Our novel contribution is that we simultaneously perform skew scheduling and tree routing so that each local skew range is incrementally refined to a skew value that minimizes the wirelength increase during the bottom-up merging phase of DME. As a result, not only is the skew schedule feasible, but also the wirelength increase is minimized at each merging step of clock tree construction. The experimental results show very encouraging improvement over the previous BST/DME algorithm on three ISCAS89 benchmarks under general skew constraints in terms of total routing wirelength.

Categories and Subject Descriptors: B.7.2 [**Integrated Circuits**]: Design Aids—*Placement and routing*; J.6 [**Computer Applications**]: Computer-Aided Engineering—*Computer-aided design (CAD)*

General Terms: Algorithms, Design, Experimentation, Performance

Additional Key Words and Phrases: Useful Skew, clock tree, merging and embedding, merging region, feasible skew range, incremental skew scheduling

1. INTRODUCTION

In a synchronous digital system, the clock signals from the clock source to the clock pins of all sequential elements can have different propagation delays through a clock distribution network. That causes clock skews among the sequential elements. The clock skew and the logic delay between two adjacent sequential elements directly determine the lower bound of the clock period,

This work was partially supported by NSF (CCR-9984553).

Authors' addresses: C.-W. A. Tsao: Celestry Design Technologies, Inc., San Jose, CA 95134; email: tsaoalbert@ieee.org; C.-K. Koh: School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907-1285; email: chengkoh@ecn.purdue.edu.

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

© 2002 ACM 1084-4309/02/0700-0359 \$5.00

and hence the upper bound of the system frequency. Due to the ever-increasing die sizes and the continued scaling of devices and interconnects, the control of clock skew in a clock distribution network is rapidly becoming a critical design problem.

High-performance clock design is an area of active research. Previous work in this area can be divided into two categories: focusing either on clock skew optimization without considering layout synthesis, or on clock routing with simplified skew constraints. Several clock scheduling techniques—including linear programming, graph-based approaches, and genetic algorithms—reported in Fishburn [1990], Sakallah et al. [1990], Szymanski [1992], Shenoy et al. [1992], Deoker and Sapatnekar [1994], Sapatnekar and Deokar [1996], Vuillod et al. [1996], Vittal et al. [1996], and Neves and Friedman [1996] belong to the first category of research on clock tree synthesis. These techniques determined the skews among clock pins that optimized, for example, performance and robustness. However, these approaches generated a clock schedule without considering its impact on the layout synthesis of the clock net.

The second category of works includes topology synthesis of the zero-skew tree (ZST) [Tsay 1993], and ZST routing based on the deferred-merge embedding (DME) algorithm [Edahiro 1992; Chao et al. 1992; Kahng and Tsao 1996]. Cong and Koh [1995], Huang et al. [1995], Kahng and Tsao [1997], and Cong et al. [1998] extended the DME algorithm to consider bounded-skew tree (BST) routing. In this article, we follow the convention used in the existing literature on clock routings: DME-based ZST routing algorithms are simply called DME, and DME-based BST routing algorithms are called BST/DME. When there is a (maximal) global skew bound $B \geq 0$ that satisfies all local skew constraints, BST/DME can be applied with this global skew bound B [Xi and Dai 1997]. However, both DME and BST/DME approaches severely limit the solution space for skew scheduling. In BST/DME, the skew range between any pair of clock pins is shrunk to the intersection of all given local skew constraints. In DME, the skew range between any clock pins is restricted to a single zero-skew value. As a result, solution quality degrades. Moreover, zero-skew and bounded-skew schedules may not always be feasible under the general skew constraints. A straightforward extension of the DME-based algorithms for the general skew constraints is to (1) calculate a feasible skew schedule using any of the approaches reported in Fishburn [1990], Sakallah et al. [1990], Szymanski [1992], Shenoy et al. [1992], Deokar and Sapatnekar [1994], Sapatnekar and Deokar [1996], Vuillod et al. [1996], Vittal et al. [1996], and Neves and Friedman [1996], and then (2) simplify the useful-skew routing as a prescribed-skew problem, for which the DME-based algorithms are readily applicable [Chao et al. 1992]. However, this is a special case of our UST/DME algorithms; we maintain the flexibility of skew scheduling throughout the entire process of the DME-based routing.

The first work that incorporated a flavor of simultaneous clock scheduling and routing was presented in Xi and Dai [1997]. The UST-BP algorithm of Xi and Dai [1997] searched for a good, easy-to-realize clock layout, called the useful skew tree (UST), under general clock skew constraints. An initial feasible skew schedule was first used to realize the initial clock tree. A simulating annealing

process was then used to improve the routing. At each step, the UST-BP algorithm perturbed the routing and verified if the resulting skew schedule was still feasible, with the verification being a nontrivial process. Clearly, the schedule did not guide the tree synthesis process; it simply pruned illegal solutions.

In this article, we investigate variants of the UST problem: the first considers the synthesis of clock layout under general skew constraints with a prescribed topology and the second without. We propose new algorithms for the simultaneous skew scheduling and routing of USTs, both of which are based on the DME or the BST/DME paradigm; hence, the names UST/DME and Greedy-UST/DME for the algorithms solving the first and second UST variants, respectively. Following the convention, we refer to both algorithms as the UST/DME approaches. Our novel contribution in this article is an incremental skew scheduler that dynamically determines the relative skew values during the process of tree layout synthesis. The advantage of our approaches is that we construct the clock layout and determine the skew schedule at the same time such that the resulting skew schedule is not only feasible, but also best for routing in terms of wirelength. The experimental results show that the UST/DME approaches are superior to the previous BST/DME algorithm on three ISCAS89 benchmarks under general skew constraints in terms of total routing wirelength.

The remainder of this article is organized as follows. In Section 2, we formulate the useful skew routing problem. In Section 3, we present a constraint graph for the capturing of skew constraints, and a graph-based approach for incremental scheduling. In Section 4, we present the UST/DME algorithm that simultaneously performs scheduling and routing with a prescribed topology. We also present the Greedy-UST/DME algorithm that synthesizes the clock skew and layout without a prescribed topology. In Section 5, we compare the UST/DME approaches to other DME-based approaches, illustrating the superiority of our approaches. Finally, we conclude in Section 6. Early partial results from this work were presented in Tsao and Koh [2000].

2. PRELIMINARIES

Consider a simple synchronous circuit using positive edge-triggered flip-flops (FFs) as the sequential elements under a single-phase clocking scheme (see Figure 1). We use $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$ to denote the set of clock pins of flip-flops in the circuit, with s_i being the clock pin of flip-flop FF_i . A pair of flip-flops are *sequentially adjacent* when only some combinational logic exists between the two flip-flops. Let FF_i and FF_j be two sequentially adjacent flip-flops, with FF_i feeding data to FF_j . Due to the unbalanced interconnect delays in the clock distribution network, clock edges may arrive at clock pins s_i and s_j at different times. Let t_i and t_j be the signal delays from the clock source (s_0) to s_i and s_j , respectively. Then, the clock skew between s_i and s_j , denoted $skew_{i,j}$, is defined to be

$$skew_{i,j} = t_i - t_j.$$

When $skew_{i,j}$ is excessively positive or negative, the data (from FF_i) are produced too late to be clocked in by FF_j (zero-clocking) or too early so they race through FF_j (double-clocking), respectively. To avoid any incorrect operations,

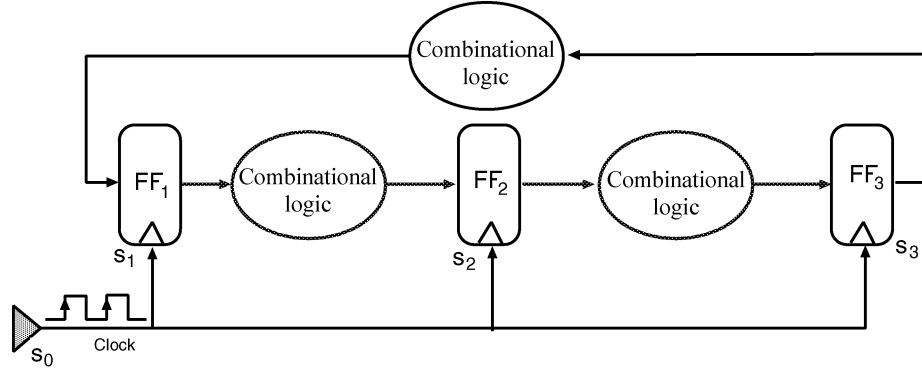


Fig. 1. A simple synchronous circuit with three positive edge-triggered flip-flops (FFs), where s_i is the clock pin of FF i and s_0 is the the clock source. The clock skew between s_i and s_j is defined by $t_i - t_j$, where t_i and t_j are the signal delays from s_0 to s_i and s_j , respectively.

we must bound $skew_{i,j}$ by the following *local* skew constraints.

$$t_i - t_j \geq t_{hold,max} - t_{pFF,min} - t_{logic,min} \quad (1)$$

$$t_i - t_j \leq C_P - t_{pFF,max} - t_{logic,max} - t_{setup,max} \quad (2)$$

where $t_{logic,max}$ and $t_{logic,min}$ are the maximum and minimum delays through the combinational logic; $t_{pFF,max}$ and $t_{pFF,min}$ are the maximum and minimum propagation delays through the flip-flop; and C_P is the clock period. For the correct latching in of data, the amount of time that the data have to remain stable before and after the clock triggers the flip-flop are t_{setup} and t_{hold} , respectively.

Designers may impose additional constraints to make circuits more robust to process variations and clock jitters, or consume less power [Xi and Dai 1996]. For example, we can make a circuit more robust to process variations and clock jitters by adding a safety margin to the lower bound constraint in Equation (1), or subtracting a safety margin from the upper bound constraint in Equation (2):

$$t_i - t_j \geq t_{hold,max} - t_{pFF,min} - t_{logic,min} + \delta_l \quad (3)$$

$$t_i - t_j \leq C_P - t_{pFF,max} - t_{logic,max} - t_{setup,max} - \delta_u \quad (4)$$

where $\delta_l \geq 0$ and $\delta_u \geq 0$ are the safety margins. In general, safety margins may vary for different pairs of flip-flops.

For simplicity, we use $l_{i,j} \leq skew_{i,j} = t_i - t_j \leq u_{i,j}$ to represent lower- and upper-bound skew constraints between s_i and s_j . For convenience, we also denote an inequality $a \leq x \leq b$ as $x \in [a, b]$. Let $R_{i,j} = [l_{i,j}, u_{i,j}]$; we use $C = \{t_i - t_j \in R_{i,j}\}$ to denote the set of skew constraints for all sequentially adjacent clock pins s_i and $s_j \in S$. A skew schedule \mathcal{X} is an assignment of delay values t_i to each clock pin s_i , and \mathcal{X} is feasible if $skew_{i,j} = t_i - t_j$ satisfies skew constraints in C ; that is, $l_{i,j} \leq skew_{i,j} \leq u_{i,j}, \forall s_i, s_j \in S$.

The problem we address in this article can be stated as follows.

Minimum-Length Useful Skew Routing Tree Problem. Given the clock pin locations of $S = \{s_1, \dots, s_n\}$ and a set of skew constraints $C = \{t_i - t_j \in R_{i,j} = [l_{i,j}, u_{i,j}]\}$, for clock pins $s_i, s_j \in S$, find a clock tree T connecting S such that the total length is minimized subject to skew constraints C .

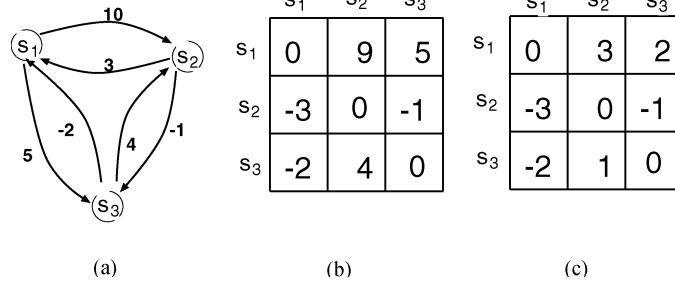


Fig. 2. An example showing the incremental skew scheduling based on an all-pairs shortest-distance matrix: (a) constraint graph G_C capturing the local skew constraints: (i) $t_1 - t_2 \in [-10, 3] = R_{1,2}$, (ii) $t_1 - t_3 \in [-5, -2] = R_{1,3}$, and (iii) $t_2 - t_3 \in [1, 4] = R_{2,3}$; (b) all-pairs shortest distance matrix \mathcal{D} constructed from G_C ; (c) all-pairs shortest distance matrix after committing the skew of s_1 and s_2 to $skew_{1,2} = -3$.

3. SKEW CONSTRAINTS AND SCHEDULING

We use a constraint graph $G_C = (V, E)$ to capture skew constraints as in Cormen et al. [1990], Deokar and Sapatnekar [1994], and Sapatnekar and Deokar [1996]. Each clock pin in \mathcal{S} corresponds to a vertex in the constraint graph. For each skew constraint $t_i - t_j \in R_{i,j} = [l_{i,j}, u_{i,j}]$, we generate two directed edges in G_C , $e_{i,j}$ and $e_{j,i}$. The former edge captures the lower-bound constraint and the latter edge the upper-bound constraint. The weight of $e_{i,j}$, denoted $w_{i,j}$, is $-l_{i,j}$, and the weight of $e_{j,i}$, denoted by $w_{j,i}$, is $u_{i,j}$. Consider the example in Figure 1 with the following local skew constraints among the three flip-flops.

- (1) $t_1 - t_2 \in [-10, 3] = R_{1,2}$,
- (2) $t_1 - t_3 \in [-5, -2] = R_{1,3}$, and
- (3) $t_2 - t_3 \in [1, 4] = R_{2,3}$.

Figure 2(a) is the corresponding constraint graph.

Interestingly, even though zero-skew is within these local skew constraints, we cannot choose $skew_{1,2} = 0 \in R_{1,2}$. Otherwise, the other two skew constraints $t_1 - t_3 \in R_{1,3}$ and $t_2 - t_3 \in R_{2,3}$ cannot be satisfied simultaneously, and thus no feasible skew schedule can be found. A careful study, presented in Section 3.1, reveals that there is an implicit (and a more stringent) skew constraint $t_1 - t_2 \in R'_{1,2} = [-9, -3]$ between s_1 and s_2 , which is transitively induced by the other skew constraints $R_{1,3}$ and $R_{2,3}$. The contribution of our work is to represent all these implicit and explicit skew constraints together for each pair of clock pins by a feasible skew range (FSR). All these FSRs, computed by an all-pairs shortest distance matrix for G_C , precisely represent the same feasible solution space as defined by the original skew constraints.

3.1 Computation of Feasible Skew Range

Consider a circuit with four flip-flops represented by clock pins $\{s_1, s_2, s_3, s_4\}$. We assume that FF_1 feeds data to FF_2 and FF_3 , both of which feed data to FF_4 . First, we consider the four upper-bound local skew constraints among the four

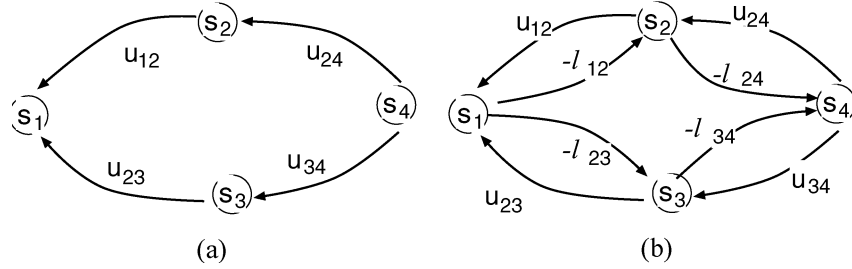


Fig. 3. An example showing constraint graph construction from given skew constraints: (a) upper bounds; (b) upper/lower bounds. (i) $t_1 - t_2 \leq u_{1,2}$, (ii) $t_2 - t_4 \leq u_{2,4}$, (iii) $t_1 - t_3 \leq u_{1,3}$, (iv) $t_3 - t_4 \leq u_{3,4}$ in (a); four more skew constraints are added in (b): (v) $l_{1,2} \leq t_1 - t_2$, (vi) $l_{2,4} \leq t_2 - t_4$, (vii) $l_{1,3} \leq t_1 - t_3$, and (viii) $l_{3,4} \leq t_3 - t_4$.

clock pins: $t_1 - t_2 \leq u_{1,2}$, $t_2 - t_4 \leq u_{2,4}$, $t_1 - t_3 \leq u_{1,3}$, and $t_3 - t_4 \leq u_{3,4}$. Although FF_1 and FF_4 are not sequentially related, there are two transitively induced constraints between s_1 and s_4 :

$$\begin{aligned} (t_1 - t_2) + (t_2 - t_4) &\leq u_{1,2} + u_{2,4}, \\ (t_1 - t_3) + (t_3 - t_4) &\leq u_{1,3} + u_{3,4}. \end{aligned}$$

Therefore, the implied upper-bound constraint between s_1 and s_4 is

$$t_1 - t_4 \leq \min\{u_{1,2} + u_{2,4}, u_{1,3} + u_{3,4}\}.$$

In the corresponding constraint graph $G_C = (V, E)$, with $V = \{s_1, s_2, s_3, s_4\}$, we add a directed edge $e_{j,i} \in E$ of weight $w_{j,i} = u_{i,j}$ from s_j to s_i for each constraint $t_i - t_j \leq u_{i,j}$ as shown in Figure 3(a). Therefore, the upper bound of the skew constraint between s_1 and s_4 is equal to the shortest distance from s_4 to s_1 .

Next, we add lower bounds to the above constraints: $l_{1,2} \leq t_1 - t_2 \leq u_{1,2}$, $l_{2,4} \leq t_2 - t_4 \leq u_{2,4}$, $l_{1,3} \leq t_1 - t_3 \leq u_{1,3}$, and $l_{3,4} \leq t_3 - t_4 \leq u_{3,4}$. Because $l_{i,j} \leq t_i - t_j$ is equivalent to $t_j - t_i \leq -l_{i,j}$, we add a directed edge $e_{i,j} \in E$ of $w_{i,j} = -l_{i,j}$ from s_i to s_j for each lower-bound skew constraint $l_{i,j} \leq t_i - t_j$ (see Figure 3(b)). Similarly, the lower bound of the transitively induced constraints between s_1 and s_4 is equal to the negative of the shortest distance from s_1 to s_4 . When G_C has no negative-weight cycles Cormen et al. [1990], the shortest distance from s_i to s_j , denoted $d_{i,j}$, is well defined. It is proved in Cormen et al. [1990] that feasible skew schedules subject to \mathcal{C} exist if and only if G_C has no negative-weight cycles.¹

Therefore, given a constraint graph G_C without negative-weight cycles, we can build an all-pairs shortest distance matrix $\mathcal{D} = \{d_{i,j} : s_i, s_j \in \mathcal{S}\}$ from G_C in $O(n^3)$ by the Floyd-Warshall algorithm [Cormen et al. 1990] to represent all $FSRs$. We also refer to matrix \mathcal{D} as the FSR matrix. The *feasible skew range* $FSR_{i,j}$ between s_i and s_j is maximally bounded by $[-d_{i,j}, d_{j,i}]$. Figure 2(b) is

¹In Cormen et al. [1990], a feasible skew schedule (if it exists) is obtained as follows. Augment $G_C = (V, E)$ by adding a source vertex s_0 with outgoing edges of zero weight to each $s_i \in V$. Then the shortest distances from s_0 to all nodes $\{d_{0,i} | s_i \in V\}$ form a feasible skew schedule satisfying constraints in \mathcal{C} .

the resultant *FSR* matrix for the original constraint graph in (a). In particular, for a bounded-skew routing with a global skew bound of $B \geq 0$ to be feasible under \mathcal{C} , the range $[-B, B]$ must be within $[-d_{i,j}, d_{j,i}]$ for all i, j . Equivalently, $B \geq 0$ exists if and only if $0 \in [-d_{i,j}, d_{j,i}]$, $\forall i, j$. In Figure 2, no feasible global skew bound exists since $0 \notin [-d_{1,2}, d_{2,1}] = [-9, -3]$.

3.2 Incremental Scheduling

In the following, we show that if we start with a constraint graph G_C without negative-weight cycles, then we do not create any negative-weight cycles in G_C when we narrow any $FSR_{i,j}$ to a subrange in $[-d_{i,j}, d_{j,i}]$. Therefore, all shortest distances after the change continue to be well defined and so do all *FSRs*. Equivalently, a feasible skew schedule subject to \mathcal{C} still exists. That is the basis of *incremental scheduling*; we incrementally refine each $FSR_{i,j}$ to a single value and a feasible skew schedule is always guaranteed.

THEOREM 1. *Suppose the constraint graph G_C constructed from skew constraints \mathcal{C} has no negative-weight cycles. Then, (i) it does not create any negative-weight cycles in G_C by shrinking $FSR_{i,j} = [-d_{i,j}, d_{j,i}]$ to a subrange $[x, y]$ such that $-d_{i,j} \leq x \leq y \leq d_{j,i}$; (ii) let $d'_{k,l}$ denote the newly updated shortest distance from node k to node l for $1 \leq k \neq l \leq n$. Then, $d'_{k,l}$, for $1 \leq k \neq l \leq n$ is well defined and can be updated as follows.*

$$d'_{k,l} = \min\{d_{k,l}, d_{k,i} - x + d_{j,l}, d_{k,j} + y + d_{i,l}\}, \quad \forall 1 \leq k \neq l \leq n.$$

Equivalently, the new $FSR_{k,l} = [-d'_{k,l}, d'_{l,k}]$ is also well defined for $1 \leq k \neq l \leq n$ exists.

PROOF. For simplicity, we assume that G_C is a complete graph; if s_i and s_j are not sequentially adjacent, then edge weights $w_{i,j}$ and $w_{j,i}$ equal ∞ . Consider narrowing $FSR_{i,j} = [-d_{i,j}, d_{j,i}]$ to a subrange $[x, y]$, where $-d_{i,j} \leq x \leq y \leq d_{j,i}$. In the following discussion, we use $w'_{i,j}$ and $w'_{j,i}$ to denote new edge weights after the change. The corresponding changes on the constraint graph G_C are the decreases of edge weights $w_{i,j}$ ($\geq d_{i,j}$) to $w'_{i,j} = \min\{-x, w_{i,j}\} = -x$ and $w_{j,i}$ ($\geq d_{j,i}$) to $w'_{j,i} = \min\{y, w_{j,i}\} = y$. The following statements hold.

- (1) The new shortest distance from s_i to s_j , $d'_{i,j}$, is now $-x$, and the shortest distance from s_j to s_i , $d'_{j,i}$, is y . Therefore, the two-node cycle linking s_i and s_j has the smallest weight among all cycles that go through s_i and s_j .
- (2) If any negative-weight cycle is created in the new constraint graph, the cycle must go through s_i and s_j since no other edge weight has changed.

From the preceding statements, we conclude that if some negative-weight cycle is created, the two-node cycle linking s_i and s_j must be a negative-weight cycle with weight equal to $-x + y < 0$. However, this contradicts the fact that $x \leq y$ (i.e., $[x, y]$ is nonempty).

Moreover, for all k, l such that $1 \leq k \neq l \leq n$, the shortest distance between s_k and s_l in the newly updated G_C may change. When that happens, the new shortest path goes through either $e_{i,j}$ or $e_{j,i}$ (but not both). Therefore, the

shortest distance $d_{k,l}$ is updated by the equation:

$$\begin{aligned} d'_{k,l} &= \min\{d_{k,l}, d_{k,i} + w'_{i,j} + d_{j,l}, d_{k,j} + w'_{j,i} + d_{i,l}\} \\ &= \min\{d_{k,l}, d_{k,i} - x + d_{j,l}, d_{k,j} + y + d_{i,l}\}. \end{aligned} \quad (5)$$

Because s_i and s_j do not form a negative-weight cycle, if the new shortest path between s_k and s_l goes through $e_{i,j}$, then the new intermediate shortest paths ($s_k \rightarrow \dots \rightarrow s_i$) and ($s_j \rightarrow \dots \rightarrow s_l$) do not go through s_j and s_i , respectively. Therefore, the new intermediate shortest paths do not go through $e_{i,j}$ or $e_{j,i}$. Equivalently, $d_{k,i}$ and $d_{j,l}$ do not change if $d_{k,l}$ is updated because of $w'_{i,j}$. Similarly, if the new shortest path between s_k and s_l goes through $e_{j,i}$, then the new intermediate shortest paths ($s_k \rightarrow \dots \rightarrow s_j$) and ($s_i \rightarrow \dots \rightarrow s_l$) do not go through s_i and s_j , respectively. Therefore, $d_{k,j}$ and $d_{i,l}$ do not change if $d_{k,l}$ is updated because of $w'_{j,i}$. In other words, the computation of new shortest distances can be carried out independently and in any arbitrary order. Therefore, the computation of the new shortest distance $d'_{k,l}$ can be carried out in a single step as shown in Equation (5), unlike the costly iterative relaxation approach needed for the computation of shortest distances by the Floyd–Warshall algorithm [Cormen et al. 1990] \square

When we narrow a nontrivial feasible skew range to a single skew value, we say that a *skew commitment* is made. The following theorem says that the $FSR_{i,j}$ is the *maximum* skew range between clock pins s_i and s_j such that committing $skew_{i,j}$ to any $x \in FSR_{i,j}$ still guarantees a feasible skew schedule.

THEOREM 2. *Suppose there exists a feasible skew schedule subject to skew constraints C . If we make a skew commitment between clock pins s_i and s_j by shrinking $FSR_{i,j} = [-d_{i,j}, d_{j,i}]$ to a trivial range $[x, x]$ to form tighter skew constraints C' , then there exists a feasible skew schedule subject to C' if and only if $x \in FSR_{i,j}$.*

PROOF.

(\Leftarrow) This is directly implied by Theorem 1.

(\Rightarrow) We prove it by contradiction. Without loss of generality, assume that we choose $x \notin FSR_{i,j} = [-d_{i,j}, d_{j,i}]$ such that $x > d_{j,i} \geq -d_{i,j}$. Then, the corresponding changes on the constraint graph G_C are: (i) edge weight $w_{i,j}$ ($\geq d_{i,j}$) decreases to $w'_{i,j} = \min\{-x, w_{i,j}\} = -x < d_{i,j}$, and (ii) $w_{j,i}$ ($\geq d_{j,i}$) remains the same as $w'_{j,i} = \min\{x, w_{j,i}\} = w_{j,i}$. Thus the new shortest distance from s_i to s_j decreases to $d'_{i,j} = \min\{-x, d_{i,j}\} = -x < d_{i,j}$, and the shortest distance from s_j to s_i is not changed (i.e., $d'_{j,i} = d_{j,i} < x$). Then the cycle that goes through the shortest paths from node i to node j and from node j to node i has a negative weight of $d'_{i,j} + d'_{j,i} = -x + d_{j,i} < 0$. Therefore, if we shrink $FSR_{i,j}$ to any value x outside $[-d_{i,j}, d_{j,i}]$, there exist no feasible schedules because of the negative-weight cycles in the new constraint graph G_C . \square

Figure 4 shows an $O(n^2)$ algorithm to update matrix D after we make a skew commitment $skew_{i,j} = x$ by shrinking $[l_{i,j}, u_{i,j}]$ to $[x, x]$. In general, the algorithm can be applied whenever we refine the skew range of a pair of clock pins. Figure 2(c) shows the resultant matrix D after we commit $skew_{1,2} = -3$: $FSR_{1,3} = [-2, -2]$ and $FSR_{2,3} = [1, 1]$.

Input: skew commit $skew_{i,j} = x$, an all-pairs shortest distance matrix $\mathcal{D} = \{d_{i,j}\}$
Output: an updated matrix \mathcal{D}
Set $d_{i,j} = -x$ and $d_{j,i} = x$ for each $d_{k,l}, 1 \leq k \neq l \leq n$ in \mathcal{D} Set $d_{k,l} = \min\{d_{k,l}, d_{k,i} - x + d_{j,l}, d_{k,j} + x + d_{i,l}\}$

Fig. 4. Procedure for updating the all-pairs shortest distance matrix after a skew commitment.

3.3 Complexity Analyses

We now construct an $O(n^3)$ algorithm for incremental scheduling. First, an all-pairs shortest path algorithm like Floyd–Warshall is used to generate matrix \mathcal{D} in $O(n^3)$ time. While there exist uncommitted skews, we select one of them and commit it. Then, we apply the incremental update algorithm. In the worst case, we have to perform $n - 1$ skew commitments; hence the $O(n^3)$ complexity. In our proposed UST/DME approaches, we select the uncommitted skew (for skew commitment) and determine the skew to be committed based on its impact on clock tree wirelength reduction.

When the constraint graph $G_C(V, E)$ is sparse such that the number of edges $m = |E|$ is linearly proportional to the number of nodes n (i.e., $m = O(n)$), we can reduce the complexity of the incremental scheduler as follows. Let $P = \{(s_{i_k}, s_{j_k}) | k = 1, \dots, n - 1\}$ denote the ordered set of $n - 1$ skew commitments, where (s_{i_k}, s_{j_k}) is the k th pair of clock pins making a skew commitment. Without using the *FSR* matrix, we can simply compute the shortest distances between nodes s_{i_k} and s_{j_k} using the Bellman–Ford algorithm [Cormen et al. 1990] to obtain FSR_{i_k, j_k} . When G_C has no negative-weight cycles and is sparse, the complexity of Bellman–Ford is in practice $O(c \cdot m) = O(n)$, where c is a small constant [Szymanski 1992, Liao and Wong 1983]. After committing to a skew value in FSR_{i_k, j_k} , we update G_C with new edges or new edge weights between nodes s_{i_k} and s_{j_k} to reflect the skew commitment. That is, the number of edges either increases by 2 or remains unchanged, depending on whether there are existing edges between s_{i_k} and s_{j_k} . After the $n - 1$ skew commitments, the number of edges will still be $O(n)$. Therefore, the complexity of the incremental scheduler due to $n - 1$ skew commitments can be reduced to $(n - 1) \times c \cdot m = O(n^2)$. On the other hand, if G_C is not sparse, the incremental scheduler complexity without the *FSR* matrix can be up to $(n - 1) \times O(nm) = O(n^4)$.

3.4 Comparisons with Related Works

To distinguish our contribution in feasible skew range computation, we particularly note the following related literature. The work in Sapatnekar and Maheshwari [1998] used a similar concept of the as-late-as-possible (ALAP) and as-soon-as-possible (ASAP) skews to speed up the retiming of large circuits. By choosing some source node, say, s_1 , from G_C , the ALAP skew of each node s_i is “defined” to be the shortest distance from s_1 to s_i in G_C , which can be

obtained by applying the Bellman–Ford algorithm to G_C .² Similarly, the ASAP skew value of node s_i is “defined” to be the negative of the shortest distance from s_i to s_1 , which can be obtained from the Bellman–Ford solution on the transpose of G_C where all edge directions are reversed.

Based on the ASAP/ALAP skews, lower and upper bounds on the retiming variables can be derived to prune the search space for the retiming problem. However, these ASAP/ALAP skews do not precisely represent all the given and transitive skew constraints; they capture only the skew constraints between the source node and nonsource nodes, but not among the nonsource nodes. In Figure 2(a), for example, if we choose s_1 as the source node, the ASAP/ALAP skews correspond to the first column and first row of the matrix in Figure 2(b). They do not capture the skew constraint between s_2 and s_3 . We can see that the skew values $skew_{1,2} = -3$ and $skew_{1,3} = -3$, which are valid skew values according to the bounds defined by ASAP and ALAP skews, constitute an infeasible skew schedule. Therefore, the bounds defined by ASAP/ALAP skews may contain infeasible skew schedules. Hence, the ASAP/ALAP skew bounds cannot replace either the given retiming constraints or the given skew constraints. On the other hand, the *FSR* matrix captures the given skew constraints completely.

The work in Neves and Friedman [1996] used a similar concept of effective permissible range for the skew scheduling to improve the circuit tolerance to the process variation. The permissible skew ranges between any pairs of clock pins are represented by linear algebraic equations that capture all the forward and backward paths between clock pins. Thus the computational complexity depends on the number of paths between clock pins, which can be exponentially large. Unlike their equation-based operations, we compute and update *FSRs* by simple matrix operations with low polynomial time-complexity for simultaneous skew scheduling and clock routing. The final skew schedule is guaranteed to be feasible since all *FSRs* are updated whenever a local skew is selected and committed.

4. THE UST/DME APPROACHES

The major merit of the proposed UST/DME approaches is the incremental skew scheduling that heuristically determines the best relative skew between clock pins for wirelength minimization. The process of tree construction follows the two-phase approach of the DME-based paradigm.

- (1) Bottom-Up Phase: Construct a binary tree of merging regions (or segments) that represent the loci of possible embedding points of internal nodes in a bottom-up order.
- (2) Top-Down Phase: Determine the exact locations of the internal nodes in a top-down order.

²The work in Sapatnekar and Maheshwari [1998] actually used longest paths to compute ALAP and ASAP skews. As the shortest and longest path problems are equivalent, we use the shortest path formulation in the discussion for ease of comparison.

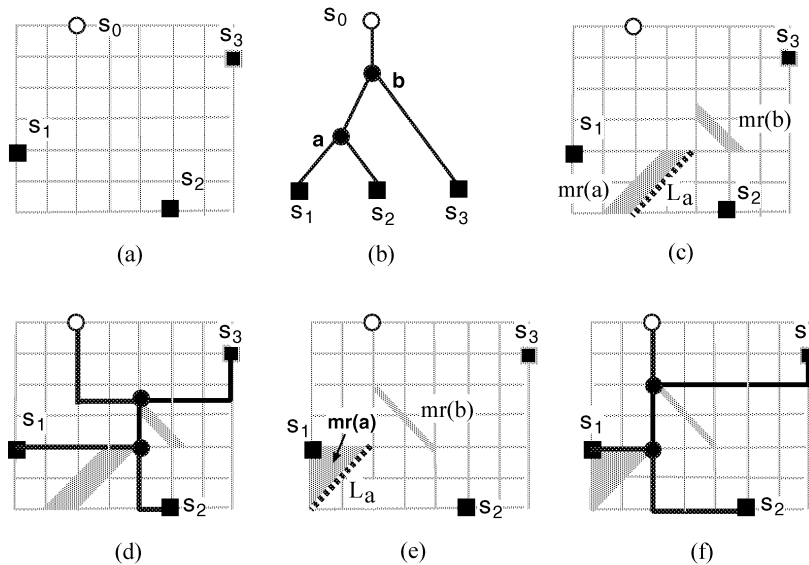


Fig. 5. An example showing the clock tree routing, by the DME-based paradigm under the linear delay model: (a) locations of three clock pins (sinks) and the clock source; (b) abstract topology \mathcal{T} for the DME algorithm; (c) tree of merging regions by BST/DME under the global skew bound $B = 1$; (d) bounded-skew tree by BST/DME for \mathcal{T} (cost = 17.5); (e) tree of merging regions by UST/DME for topology \mathcal{T} subject to the skew constraints given in Figure 2; (f) useful-skew tree by UST/DME for \mathcal{T} (cost = 17). Note that the joining segments labeled as $L_a \subseteq mr(a)$ in (c) and (e) are used for constructing the parent merging region $mr(b)$.

Bottom-Up Phase. We first distinguish a physical interconnect tree T from its *abstract topology* \mathcal{T} . An abstract topology is a binary tree (with the exception that the root node s_0 has only one child) such that all clock pins are the leaf nodes of the binary tree. T is an embedding of the abstract topology \mathcal{T} ; that is, each internal node $v \in \mathcal{T}$ is mapped to a location $l(v)$ in the Manhattan plane. Node $v \in \mathcal{T}$ is connected to its parent by edge e_v , and the cost of edge e_v is its wirelength, denoted $|e_v|$. The cost of \mathcal{T} , denoted $cost(\mathcal{T})$, is equal to the sum of its edge costs. Figure 5 shows the physical layout of the example in Figure 2 and an abstract topology.

The DME algorithm (for ZST, BST, or UST) requires an abstract topology \mathcal{T} that gives the order of merging in the physical tree. Let T_v denote the physical tree rooted at node v . If v is the parent node of u and w in \mathcal{T} , then T_u and T_w are constructed before they are merged at v to form T_v ; hence the bottom-up processing order. In Figure 5, for example, the order of merging is: T_{s_1} merges with T_{s_2} at parent node a , and T_a merges with T_{s_3} at b , which is connected to the root node directly. On the other hand, the Greedy-DME algorithm (for ZST, BST, or UST) constructs the abstract topology from the physical layout process.

Associated with each node $v \in \mathcal{T}$ is a merging region, denoted $mr(v)$, which allows T_u and T_w to be merged at v with minimum added wire $|e_u| + |e_w|$ subject to the relevant skew constraints. For ZST construction, it is proved that under the linear or Elmore delay model, the merging region must be a *Manhattan arc* (a segment with slope 1 or -1) [Edahiro 1992; Chao et al. 1992].

Input: set of clock pins \mathcal{S} , local skew constraints \mathcal{C} , abstract topology \mathcal{T}
Output: A clock tree routing T satisfying constraints in \mathcal{C} or no solution
<ol style="list-style-type: none"> 1. Construct constraint graph $G_C = (V, E)$ from \mathcal{C} 2. If G_C has negative-weight cycles, return no solution 3. Build an all-pairs shortest distance matrix \mathcal{D} from G_C 4. for each merging of subtrees T_u and T_w to form T_v based on bottom-up topological sort of abstract topology \mathcal{T} 5. if T_u is not a clock pin, i.e., with children T_1 and T_2 6. Select $L_u \in mr(u)$ that is closest to $mr(w)$ 7. Pick clock pins $s_i \in T_1$ and $s_j \in T_2$ 8. Compute $skew_{i,j} = x$ when u is embedded in L_u 9. Update \mathcal{D} for skew commitment $skew_{i,j} = x$ (see Figure 4) 10. if T_w is not a clock pin 11. Reconstruct $mr(w)$ if \mathcal{D} is updated 12. Perform steps 6 through 9 with index u and w interchanged 13. Get a $FSR_{i,j} = [-d_{i,j}, d_{j,i}]$ for $s_i \in T_u, s_j \in T_w$ 14. Construct merging region $mr(v) \in SDR(L_u, L_w)$ with feasible skew $skew_{i,j} \in FSR_{i,j}$ 15. Perform DME top-down embedding (embed each node v in $L_v \in mr(v)$ in top-down order)

Fig. 6. The UST/DME algorithm.

For BST construction with a global skew bound $B > 0$, the merging region is (1) an octilinear polygon bounded by Manhattan arcs and rectilinear (vertical or horizontal) line segments under the linear delay model, or (2) a simple convex polygon with $O(n)$ boundary segments under the Elmore delay model, where n is the number of clock pins Cong et al. [1998]. Figure 5(c) shows the tree of the merging regions by BST/DME for the global skew bound $B = 1$. Note that the BST in Figure 5(c) is given only for illustration purposes because there exist no feasible bounded skew solutions for that example (see Section 3.1).

The underlying concept of the merging region for UST construction is the same as that of BST construction: there is a range of feasible skew values for the merging of two child nodes. We present the computation of a UST merging region in Section 4.2.

Top-Down Phase. Given the tree of merging regions, each node $v \in \mathcal{T}$ is embedded, in a top-down order, at any location $l(v) \in mr(v)$ that is at distance $|e_v|$ or less from the embedding location of the parent node of v . Figure 5(d) shows the embedding of the tree of merging regions in Figure 5(c). Because the top-down phase of UST/DME is identical to those presented in Chao et al. [1992] and Cong et al. [1998], we focus only on the bottom-up merging region construction in the remainder of this section.

4.1 The UST/DME Algorithm

The outline of the UST/DME algorithm is given in Figure 6. At the beginning of the algorithm, we have a forest of singleton trees, each containing a pin in \mathcal{S} ; we

denote the singleton tree containing s_i by T_{s_i} . The order of subtree merging is prescribed by topology \mathcal{T} in a bottom-up fashion. To determine the skew value for wirelength minimization at each DME merging step, we need to compute and store the *FSRs* for all pairs of clock pins. Therefore, before any DME merging step, we construct a constraint graph G_C from the given skew constraints \mathcal{C} , and then build an all-pairs shortest distance matrix $\mathcal{D} = \{d_{i,j} : s_i, s_j \in S\}$ from G_C to represent all *FSRs*. When the forest has more than two trees, we repeatedly select two trees, say, T_u and T_w , for merging according to the order imposed by the abstract topology \mathcal{T} , and denote the newly formed tree by T_v . In other words, we construct merging region $mr(v)$ based on $mr(u)$ and $mr(w)$, the merging regions of the child nodes of v . The computation of the parent merging region from two child merging regions is presented next.

4.2 Merging Region Construction

In the BST/DME approaches, the global skew bound B can be viewed as a local skew constraint that limits only the difference between the maximum and minimum delays from each subtree root to its clock pins. As a result, this simplified skew constraint greatly facilitates the computation of merging regions, which is stated as follows. When we merge T_u and T_w to form T_v , we first determine the closest boundary segments of child merging regions $mr(u)$ and $mr(w)$. The two closest segments $L_u \subseteq mr(u)$ and $L_w \subseteq mr(w)$, which are used to construct $mr(v)$, are called *joining segments* Cong et al. [1998]. They can either be a Manhattan arc or a rectilinear line segment. Under the linear or Elmore delay models, the skew values are always constant on Manhattan arcs and a piecewise linear function of locations on rectilinear line segments. With such properties, the “feasible skew range” (subject to the global skew bound B) between clock pins in T_u and T_w can be easily computed based on the maximum and minimum delays from L_u and L_w to clock pins in T_u and T_w , respectively. Let the shortest distance region $SDR(L_u, L_w)$ between L_u and L_w be a set of points with the minimum sum of distances to L_u and L_w . To minimize the added wirelength for merging with feasible skew, merging region $mr(v)$ is constructed within $SDR(L_u, L_w)$ such that all skews between clock pins in T_u and T_w satisfy the global skew bound B . For the example in Figure 5(c), merging region $mr(a)$ is constructed with $SDR(s_1, s_2)$ such that the skew from any points in $mr(a)$ to s_1 and s_2 is at most equal to $B = 1$. Similarly, merging region $mr(b)$ is constructed within $SDR(L_a, s_3)$, where the joining segment L_a is the boundary segment of $mr(a)$ that is closest to s_3 . The skew from any points of merging region $mr(b)$ to all three clock pins is also at most equal to $B = 1$. More details of the construction rules of merging regions can be found in Cong et al. [1998].

Although the underlying concept of the merging region in our UST/DME approaches is the same as that in the BST/DME algorithm, the computation of merging regions under incremental scheduling introduces a problem not encountered before. The problem, which relates to the fact that a skew commitment in one subtree affects the feasible skew ranges of clock pins in another, manifests itself in two ways. First, the selection of joining segments from the merging regions of child nodes is nontrivial. Selection of a joining segment L_u

from $mr(u)$ implies a commitment to some skew values for clock pins in T_u . That commitment affects the feasible skew range for clock pins in T_w .

Second, the skew values on rectilinear joining segments L_u are not constant. Each embedding point of node u on L_u means a different choice of skew commitment for clock pins in T_u . Thus, embedding u at different locations on L_u will have different impacts on the feasible skew ranges for clock pins in the other subtree. This further complicates the merging region construction under various local skew constraints. Despite these difficulties, UST/DME still outperforms the straightforward application of BST/DME for embedding a given topology because of its simultaneous topology embedding and skew scheduling (see Section 5).

To overcome the difficulties highlighted in the preceding discussion, we restrict the joining segments in the UST/DME algorithm to be Manhattan arcs only. The advantage of such a restriction is that $skew_{i,j}$ for any pair of clock pins s_i and s_j in the subtree (rooted by the Manhattan arc) is a constant value, say, $x \in FSR_{i,j}$ Cong et al. [1998]. Let L_u be such a Manhattan arc in $mr(u)$ such that it is closest to $mr(w)$. Let s_i and s_j be clock pins in T_u . Since a skew commitment of $skew_{i,j} = x$ changes $FSR_{k,l}$ for clock pins s_k and s_l in T_w , we recompute the merging region $mr(w)$ based on the updated $FSR_{k,l}$. In order to minimize the merging wirelength, another joining segment L_w is then selected from the recomputed $mr(w)$ such that it is closest to L_u . Finally, $mr(v)$ is then constructed within $SDR(L_u, L_w)$ such that the skew between $s_i \in T_u$ and $s_k \in T_w$ is feasible, that is, $skew_{i,j} \in FSR_{i,j}$.

Therefore, at the end of Steps 4 through 14, UST/DME produces a binary tree of joining segments. The joining segments L_u of node u and L_w of w are, respectively, within the merging regions $mr(u)$ and $mr(w)$, and they are determined when we compute the merging region of the parent node v of u and w .

Figures 5(e) and (f) show the tree of merging regions and the resulting tree routing that correspond to the skew commitment $skew_{1,2} = -3$ in Figure 2(c). First, merging region $mr(a)$ (shaded) is constructed within the shortest distance region $SDR(s_1, s_2)$ with s_1 and s_2 being the joining segments. The skew from any points in $mr(a)$ to s_1 and s_2 falls into the range $[-7, -3] \subset FSR_{1,2} = [-9 - 3]$, and is feasible. Next, $mr(b)$ is constructed within $SDR(L_a, s_3)$ where joining segments $L_a \in mr(a)$ is the boundary segment of $mr(a)$ that is closest to s_3 . Note that in general if s_3 is not a singleton tree of a clock pin, then $mr(s_3)$ has to be recomputed based on the updated FSR matrix after committing the skew that corresponds to the chosen joining segments $L_a \in mr(a)$. The skew schedule realized by the UST/DME tree under the linear delay model is $\{t_1 = 6, t_2 = 9, t_3 = 8\}$, which is feasible subject to the constraints given in Figure 2. On the other hand, the BST/DME tree not only produces an infeasible skew schedule, but also uses more wirelength than the UST/DME tree.

4.3 The Greedy-UST/DME Algorithm

When no prescribed topology is given, we apply the Greedy-UST/DME algorithm to determine the topology of the tree of merging regions (or joining segments) in a greedy bottom-up fashion as in Edahiro [1993]. The outline

of Greedy-UST/DME is similar to that of UST/DME except for Step 4, which is amended as follows.

4. **for** each merging of subtrees T_u and T_w to form T_v based on the nearest neighbor graph [Edahiro 1993]

As mentioned earlier, we have a forest \mathcal{F} of singleton trees at the beginning of the algorithm. The order of subtree merging is determined from the so-called nearest neighbor graph (NNG) [Edahiro 1993], which stores the promising merging pairs of subtrees based on the estimated merging cost. The algorithm first constructs an NNG that maintains the nearest neighbors (in terms of merging cost) of all merging regions of subtree roots in \mathcal{F} . In the NNG, the vertices are root nodes of \mathcal{F} , and the edges represent nearest neighbors among these root nodes. In one iteration, $|\mathcal{F}|/k$ independent nearest-neighbor pairs from the NNG are selected in a nondecreasing order of merging cost for merging, where k is a constant ranging from 2 to 4 [Edahiro 1993]. The NNG is reconstructed after each iteration. The number of subtrees is reduced to $1 - (1/k)|\mathcal{F}|$ after each iteration. So the algorithm will finish in $\log_{k/(k-1)} n$ iterations, where n is the number of clock pins.

Similar to the Greedy-DME approaches in the literature [Edahiro 1992, 1993], Greedy-UST/DME merges two subtrees at their root nodes. In contrast, BST/DME allows merging at nonroot nodes by dynamically changing subtree topologies before the merging is performed. The feature of dynamically changing subtree topologies allows BST/DME to produce very low-cost solutions when the skew bound is large. Because our incremental scheduler cannot uncommit skew commitments made in the previous merging steps without rebuilding the *FSR* matrix, topologies of subtrees in Greedy-UST/DME, once they are constructed, cannot be dynamically changed as in the BST/DME approach. In spite of such a limitation, Greedy-UST/DME still outperforms BST/DME for topology generation when the skew constraints are tight (see Section 5).

4.4 Complexity Analyses

Without the incremental scheduler, UST/DME and Greedy-UST/DME have the the same run-time complexity as original BST/DME and Greedy-BST/DME approaches, which are $O(n)$ and $O(n \log n)$ [Cong et al. 1998], respectively. With the incremental scheduler, both UST/DME and Greedy-UST/DME first take $O(n^3)$ time to build the all-pairs shortest distance matrix \mathcal{D} , and then make $n - 1$ skew commitment, each of which takes $O(n^2)$ time to update matrix \mathcal{D} . So the overall complexity of UST/DME and Greedy-UST/DME is $O(n^3)$.

As mentioned in Section 3.3, given P , the set of $(n - 1)$ pairs of clock pins arranged according to the order in which skew commitments are made, the complexity of the incremental scheduler can be reduced to $O(n^2)$ for a sparse graph by using the Bellman–Ford algorithm instead of the *FSR* matrix. For UST/DME, P is prescribed by the given abstract topology. Hence, the complexity of UST/DME can be reduced to $O(n^2)$ for a sparse constraint graph.

By a similar technique, the complexity of Greedy-UST/DME can be reduced as follows when G_C is sparse. Initially, the constraint graph G_C has n nodes,

Table I. Benchmark Circuits from Xi and Dai [1997]

Circuit	Number of Clock Pins	Number of Skew Constraints	Maximum Absolute Skew Bound (ns)
s1423	74	78	1.4
s5378	179	175	0.3
s15850	597	318	0.2

which correspond to the n singleton trees. We use the bucket decomposition method of Edahiro [1994] to construct the nearest neighbor graph (NNG), which stores $O(n)$ most promising pairs of subtrees in terms of the estimated merging cost. The estimation of each merging cost requires an *FSR* computation, which can be done in $O(n)$ by running the Bellman–Ford algorithm. Therefore, by using Bellman–Ford instead of the *FSR* matrix, the NNG can be constructed in $O(n^2)$ time. In each iteration, $n/k = O(n)$ nearest-neighbor pairs of trees are selected from the NNG for merging with skew commitments, where $2 \leq k \leq 4$ is a constant. Therefore, the overall time complexity of the first iteration, denoted T_1 , is of the same order as that of UST/DME, that is, $T_1 = O(n^2)$. Without loss of generality, we let $k = 2$. In the second iteration, there are $n/2$ trees remaining, and the number of edges in the updated G_C is still $O(n)$. Therefore, the time complexity of the second iteration, denoted $O(T_2)$, is also of the same order as that of the first iteration; that is, $O(T_2) = O(T_1)$. After $\log_2 n$ iterations, Greedy-UST/DME terminates with only one tree being left. Therefore, the overall runtime of Greedy-UST/DME for a sparse graph is

$$\sum_{i=1}^{\log n} T_i = \sum_{i=1}^{\log n} T_1 = T_1 \cdot \log n = O(n^2 \log n).$$

However, the complexity of Greedy-UST/DME is $O(n^4 \log n)$ for a dense constraint graph if we do not construct the *FSR* matrix.

5. EXPERIMENTS

The proposed UST/DME and Greedy-UST/DME algorithms are implemented in C++ and tested on three ISCAS89 benchmark circuits from Xi and Dai [1997] on Sun UltraSPARC-II machines. Table I gives the pin counts, the distribution of skew constraints, and the maximum allowable absolute skew bounds of the three benchmark circuits. We note that the corresponding constraint graphs for all three circuits are sparse with the number of edges linearly proportional to that of nodes.

The Elmore delay model is used to measure the wire delays for ease of comparison with earlier DME-based approaches. Note that it is possible to consider a higher-order delay model. The higher-order moment-based technique employed in Sapatnekar and Leither [1998] can be extended to consider UST construction. We set the k parameter for the Greedy-UST/DME algorithm to 2; that is, in each iteration, $\mathcal{F}/2$ independent nearest-neighbor pairs in the NNG are merged. We compare both UST/DME algorithms with the BST/DME algorithm [Cong et al. 1998] and the UST-BP algorithm [Xi and Dai 1997].

Table II. Comparison of Wirelengths

Circuit	ZST Wire	BST/DME		UST-BP Reduction over ZST
		Wire	Reduction over ZST	
s1423	107277	65207	39%	23%
s5378	176517	107546	39%	21%
s15850	448599	274901	39%	16%

Circuit	<i>Greedy-UST/DME</i>			<i>UST/DME</i> Width BST/DME Topology		
	Wire	Reduction over ZST	CPU Time (min:sec)	Wire	Reduction over ZST	CPU Time (min:sec)
s1423	89189	17%	0:14	67843	37%	0:01
s5378	149391	15%	1:09	112513	36%	0:08
s15850	355561	20%	9:23	265104	41%	4:48

For the UST-BP algorithm, we report only the wirelength reduction over the zero-skew routing but not the actual wirelength because of discrepancy between wirelength units. The results are summarized in Table II. For the UST/DME algorithm, we use topologies generated by the BST/DME approach [Cong et al. 1998].

Greedy-UST/DME is better than UST-BP for the largest test case s15850, whereas UST-BP performs better for s1423 and s5378. UST/DME and BST/DME have similar results since they share the same topologies, and both of them outperform Greedy-UST/DME and UST-BP significantly. Since the topology generation in Greedy-UST/DME is more restricted than in BST/DME (i.e., subtree topology cannot be changed dynamically), Greedy-UST/DME solutions are inferior to BST/DME solutions when skew bounds are relatively large. Indeed, all test cases have large maximum allowable skew bounds as shown in Table I. As a result, the BST/DME and UST/DME solutions are very close to minimal Steiner trees due to the large skew bounds.

To make a more interesting comparison under the tighter skew constraints, we also compare the Greedy-UST/DME and UST/DME algorithms with the BST/DME algorithm on the largest benchmark s15850, as shown in Figure 7, for different safety margins $\delta = \delta_l = \delta_h$ (see Equations (3) and (4)), ranging from 0 to 250 pico-seconds (ps).³ For each safety margin $\delta \in [0, 200]$ ps, the global skew bound for BST/DME is $B = (200 - \delta)$ ps, obtained by taking the intersection of all feasible skew ranges. There are no bounded-skew solutions when the safety margin δ is beyond 200 ps.⁴

Again, for smaller safety margins, Greedy-UST/DME solutions are inferior to those of BST/DME because the corresponding skew bounds for BST/DME are relatively large. However, for larger safety margins (≥ 190 ps), which are more desirable for high-performance circuits, BST/DME has little freedom of

³Because the UST-BP code is unavailable, we cannot make a similar comparison with their results.

⁴We can also obtain the prescribed-skew DME solutions for the safety margin beyond 200 ps. However, the total wirelengths of these solutions are much larger than those of the zero-skew DME solutions. For the sake of clarity, we ignore the prescribed-skew DME solutions in our comparison.

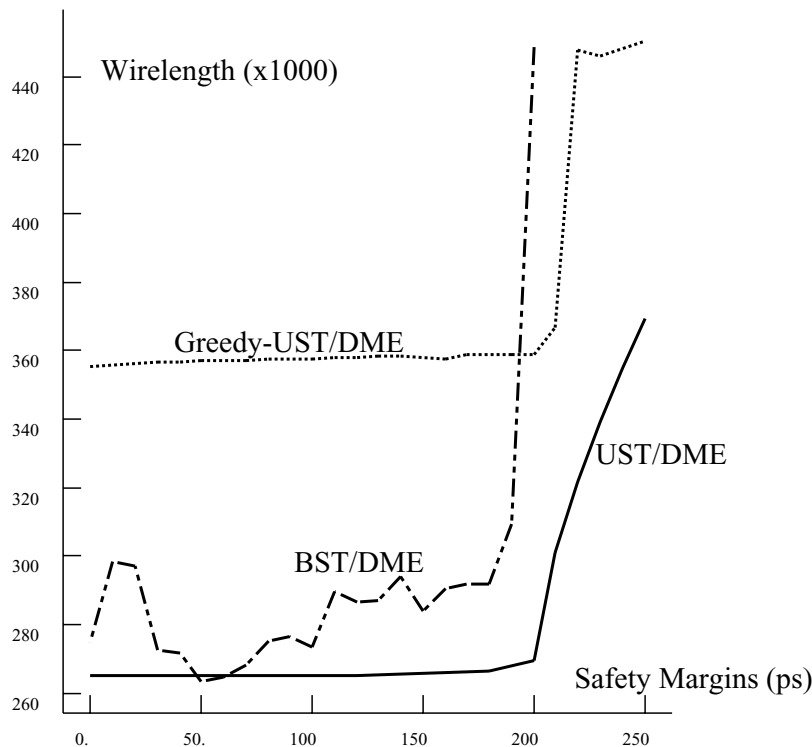


Fig. 7. Comparison of total wirelengths among BST/DME, Greedy-UST/DME, and UST/DME (with a BST/DME topology) on ISCAS89 s15850 test case for different safety margins.

skew choices (skew bound $B \leq 10$ ps) for merging subtrees. On the other hand, UST/DME can merge most of the subtrees with large feasible skew ranges since most of the local skew constraints are actually very loose (the skew scheduling is in practice limited by only a few critical local skew constraints). Therefore, Greedy-UST/DME performs better than BST/DME when safety margins are large. Also, for all of the safety margins Greedy-UST/DME solutions are at least as good as the zero-skew solution, which has the minimum freedom of skew choices for merging subtrees. Furthermore, we run the UST/DME algorithm using the abstract topology obtained by BST/DME for a loose skew bound of 150 ps. UST/DME shows clear superiority over BST/DME because of its simultaneous topology embedding and skew scheduling. We also performed the same experiments for the other two circuits. The results show similar trends as shown in Figure 7. That is, for larger safety margins, both Greedy-UST/DME and USE/DME with given topology are superior to BST/DME in terms of wirelength.

We note that the wirelengths of UST/DME solutions increase rapidly when the safety margin δ approaches 250 ps. This is due to the considerable amount of detour wire used to achieve very large skew values for some mergings of subtrees. Instead of using long detour wire, a more practical approach to achieving merging with a large skew value is to use repeaters. Moreover, repeaters are

Table III. Switch-Level Linear RC Model of the Repeater^a

Maximum Capacitive Load (C_{\max_load})	Input Pin Capacitance	Output Resistance (R_d)	Intrinsic Delay (t_{int})
0.3 pF	0.014 pF	500 Ω	90 ps

^aCell delay of the repeater is computed as $R_d \times C_L + t_{int}$, where $C_L \leq C_{\max_load}$ is the lumped capacitive load driven by the repeater.

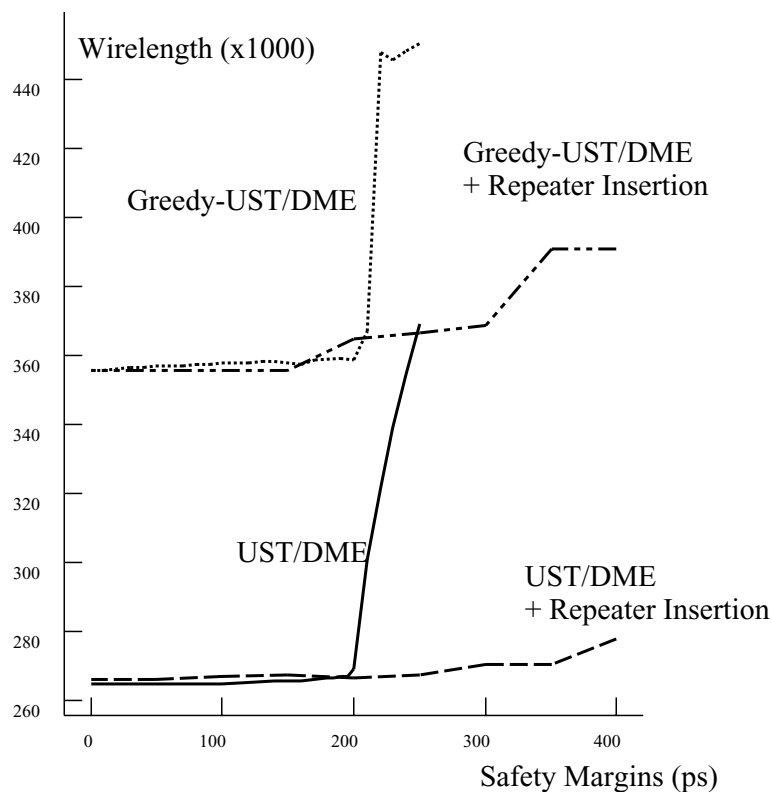


Fig. 8. Wirelength reduction by enhancing Greedy-UST/DME and UST/DME with repeater insertion on ISCAS89 s15850 test case for different safety margins.

necessary to maintain reasonable transition time in the practical clock designs. Therefore, we extend our algorithm to consider repeater insertion, in which the delays caused by repeaters are estimated based on a switch-level linear RC model. The parameters of the repeater model are shown in Table III. There are two scenarios in which a repeater is inserted: it is inserted at the root of a subtree in order to limit the capacitive load to be within the drive capability of the repeater, and, it is inserted whenever it minimizes the merging cost, which is defined to be $L_{wire} + N_{buf} \times weight$, where L_{wire} and N_{buf} are the wirelength and the number of the repeaters that are added for merging the subtrees. In other words, we perform a trade-off between repeater count and wirelength. In our experiments, we set $weight$ to be 1000. As shown in Figure 8, we can

achieve safety margins up to 400 ps (which is the limit of the circuit logic) with a reasonable amount of wirelength by using repeater insertion.

6. CONCLUSIONS

In this article, we propose an incremental skew scheduler to guide the DME-based clock routing subject to various local skew constraints. Compared to the previous BST/DME algorithm, our experiments show very encouraging results. However, for very loose skew constraints, UST/DME solutions are not as competitive as BST/DME solutions, which generally are topologies similar to minimal Steiner trees when skew bounds are large. We are seeking to further improve our incremental skew scheduler that allows more flexible adjustments of subtree topologies before the merging of subtrees.

The high complexity of our skew scheduler could be a major concern for clock designers in industry. By using the sparse constraint graph property, the time-complexities of UST/DME and Greedy-UST/DME can be improved to $O(n^2)$ and $O(n^2 \log n)$, respectively. Still, further speedup of our algorithm is possible. For example, one possible speedup is to reduce the size of the constraint graph for the incremental scheduler by applying prescribed skews to some of the nodes. However, the speedup is achieved at the expense of solution quality as such an approach may reduce the solution space for skew scheduling.

We plan to incorporate higher-order delay models in our clock tree synthesis algorithm in a future work. Also, our simple repeater insertion scheme can be improved by considering multiple choice of repeaters or the buffer sliding techniques in Kahng and Tsao [1997]. Lastly, we note that the topologies determined by our algorithm for large safety margins can be more unbalanced than those in the zero-skew routing. The unbalanced tree structures have an adverse impact on the achieved safety margins when there are process variations on the buffer parameters and wire width. This is a very challenging issue that we need to consider to make our algorithms more applicable to the practical clock designs in industry.

ACKNOWLEDGMENTS

We are grateful to Professor Sachin Sapatnekar of the University of Minnesota for the numerous discussions on the concept of ASAP/ALAP skews and the complexity of the Bellman–Ford algorithm for the sparse graph. We also thank the anonymous reviewers for their constructive comments.

REFERENCES

- CHAO, T.-H., HSU, Y.-C. H., HO, J.-M., BOESE, K. D., AND KAHNG, A. B. 1992. Zero skew clock routing with minimum wirelength. *IEEE Trans. Circ. Syst.* 39, 11 (Nov.), 799–814.
- CONG, J. AND KOH, C.-K. 1995. Minimum-cost bounded-skew clock routing. In *Proceedings of the IEEE International Symposium on Circuits and Systems* (April), 1.215–1.218.
- CONG, J., KAHNG, A. B., KOH, C.-K., AND TSAO, C.-W. A. 1998. Bounded-skew clock and Steiner routing. *ACM Trans. Des. Autom. Electron. Syst.* 3, 3, 341–388.
- CORMEN, T. H., LEISERSON, C. E., AND RIVEST, R. L. 1990. *Introduction to Algorithms*. Chapter 25.5, McGraw-Hill, New York, 539–543.

- DEOKAR, R. B. AND SAPATNEKAR, S. S. 1994. A graph-theoretic approach to clock skew optimization. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, 407–410.
- EDAHIRO, M. 1992. Minimum path-length equi-distant routing. In *Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems* (December), 41–46.
- EDAHIRO, M. 1993. A clustering-based optimization algorithm in zero-skew routing. In *Proceedings of the Design Automation Conference* (June), 612–616.
- EDAHIRO, M. 1994. An efficient zero-skew routing algorithm. In *Proceedings of the Design Automation Conference* (June), 375–380.
- FISHBURN, J. P. 1990. Clock skew optimization. *IEEE Trans. on Comput.* 39, 7 (July), 945–951.
- HUANG, J. H., KAHNG, A. B., AND TSAO, C.-W. A. 1995. On the bounded-skew routing tree problem. In *Proceedings of the Design Automation Conference* (June), 508–513.
- KAHNG, A. B. AND TSAO, C.-W. A. 1996. Planar-DME: A single-layer zero-skew clock tree router. *IEEE Trans. Comput. Aid. Des. Integ. Circ. Syst.* 15, 1 (Jan.), 8–19.
- KAHNG, A. B. AND TSAO, C.-W. A. 1997. Practical bounded-skew clock routing. *J. VLSI Sig. Process. (Special Issue on High Performance Clock Distribution Networks)* 16, 2–3 (June–July), 199–215.
- LIAO, Y.-Z. AND WONG, C. K. 1983. An algorithm to compact a VLSI symbolic layout with mixed constraints. *IEEE Trans. Comput. Aid. Des. Integ. Circ. Syst.* 2, 2 (Feb.), 62–69.
- NEVES, J. L. AND FRIEDMAN, E. G. 1996. Optimal clock skew scheduling tolerant to process variations. In *Proceedings of the Design Automation Conference*, 623–628.
- SAKALLAH, K. A., MUDGE, T. N., AND OLUKOTUN, O. A. 1990. checkTc and minTc: Timing verification and optimal clocking of synchronous digital circuits. In *Proceedings of the International Conference on Computer Aided Design*, 552–555.
- SAPATNEKAR, S. S. AND DEOKAR, R. B. 1996. Utilizing the retiming-skew equivalence in a practical algorithm for retiming large circuits. *IEEE Trans. Comput. Aid. Des. Integ. Circ. Syst.* 15, 10 (Oct.), 1237–1248.
- SAPATNEKAR, S. S. AND LEHTER, D. 1998. Moment-based techniques for rlc clock tree construction. *IEEE Trans. Circ. Syst. II: Analog Digital Sig. Process.* 45, 1 (Jan.), 69–79.
- SAPATNEKAR, S. S. AND MAHESHWARI, M. 1998. Efficient retiming of large circuits. *IEEE Trans. VLSI Syst.* 6, 1, 74–83.
- SHENOY, N., BRAYTON, R. K., AND SANGIOVANNI-VINCENTELLI, A. L. 1992. Graph algorithms for clock schedule optimization. In *Proceedings of the International Conference on Computer Aided Design*, 132–136.
- SZYMANSKI, T. 1992. Computing optimal clock schedules. In *Proceedings of the Design Automation Conference*, 399–404.
- TSAO, C.-W. A. AND KOH, C.-K. 2000. UST/DME: A clock tree router for general skew constraints. In *Proceedings of the International Conference on Computer Aided Design*, 400–405.
- TSAY, R.-S. 1993. An exact zero-skew clock routing algorithm. *IEEE Trans. Comput. Aid. Des. Integ. Circ. Syst. CAD-12*, 2 (Feb.), 242–249.
- VITTAL, A., HA, H., BREWER, F., AND MAREK-SADOWSKA, M. 1996. Clock skew optimization for ground bounce control. In *Proceedings of the International Conference on Computer Aided Design* (November), 395–399.
- VULLOD, P., BENINI, L., BOGLIOLO, A., AND DE MICHELI, G. 1996. Clock-skew optimization for peak current reduction. In *Proceedings of the International Symposium on Low Power Electronics and Design* (Aug.), 265–270.
- XI, J. G. AND DAI, W. W.-M. 1996. Jitter-tolerant clock routing in two-phase synchronous systems. In *Proceedings of the International Conference on Computer Aided Design*, 316–320.
- XI, J. G. AND DAI, W. W.-M. 1997. Useful-skew clock routing with gate sizing for low power design. *J. VLSI Sig. Process. Syst.* 16, 2/3, 163–170.

Received June, 2000; accepted March, 2002