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

Chung-Wen Albert Tsao
Ultima Interconnect Technology
Sunnyvale, CA 94089, USA
tsao@ultimatech.com

Cheng-Kok Koh
ECE, Purdue University
West Lafayette, IN 47907-1285, USA
chengkoh@ecn.purdue.edu

ABSTRACT

In this paper, we propose new approaches for solving the useful-skew tree (UST) routing problem [17]: 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 zero-skew tree [5; 1] and bounded-skew tree [8; 2] routings; hence, the names UST/DME and Greedy-UST/DME for our algorithms. They perform simultaneously skew scheduling and tree routing such that each local skew range is incrementally refined to the best skew value that minimizes the wirelength during the bottom-up merging phase of DME. The resulting skew schedule is not only feasible, but is also best for routing in terms of wirelength. 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 wirelength.

1. INTRODUCTION

Due to the ever increasing die sizes and the continued scaling of devices and interconnects, the control of clock skew caused by unbalanced wire delays in a clock distribution network is rapidly becoming a critical design problem. High-performance clock design is an area of active research. Previous works in this area can be divided into two categories; they emphasized either clock skew optimization without considering layout synthesis or clock routing with simplified skew constraints.

Several clock scheduling techniques—including linear programming, graph-based approaches, and genetic algorithms—reported in [7; 10; 12; 11; 4; 15; 14; 9] 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 of the clock net.

The second category of works includes topology synthesis of zero-skew tree (ZST) [13], and ZST routing based on the deferred-merge embedding (DME) algorithm [5; 1]. In [8; 2], the DME algorithm was extended to consider bounded-skew tree (BST) routing. In this paper, we refer to DME-based ZST routing algorithms simply as DME, and DME-based BST routing algorithms as BST/DME. However, both DME and BST/DME approaches severely limit the solution space for skew scheduling. In BST/DME, every local skew range is shrunk to a much smaller range. In DME, every local skew range is restricted to a single skew value.[1] As a result, solution quality degrades. Moreover, zero-skew and bounded-skew schedules may not always be legitimate.

The UST-BP algorithm of [17] was the first work that incorporated a "flavor" of simultaneous clock scheduling and routing. It searched for a good, easy-to-realize clock layout, called the useful skew tree (UST), under general clock skew constraints. First, it realized an initial clock tree under an initial legitimate skew schedule. Next, it improved the routing through simulated annealing. At each step, it perturbed the routing and verified whether the resulting skew schedule was still feasible. Clearly, the schedule did not guide the tree synthesis process; it simply pruned illegal solutions.

In this paper, we investigate two 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 two 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. We refer to both algorithms as the UST/DME approaches. Our novel contribution is an incremental skew scheduler that dynamically determines *feasible* relative skew values under the general skew constraints during the process of DME-based clock 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 is 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 wirelength.

The remainder of this paper is organized as follows. In Section 2, we formulate the useful skew routing problem. In Section 3, we present a graph-based approach for incremental scheduling. In Section 4, we present UST/DME and Greedy-UST/DME. In Section 5, we compare the UST/DME approaches to other approaches. Finally, we conclude the paper in Section 6.

2. PRELIMINARIES

Consider a simple synchronous circuit using positive edge-triggered flip-flops (FF's) as the sequential elements under a single-phase clocking scheme. We use $\mathcal{S} = \{s_1, s_2, \cdots, s_n\}$ 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 t_i and t_j, respectively. Then, the clock skew between s_i and s_j is defined to be $skew_{i,j} = t_i - t_j$.

*Partially supported by NSF CAREER award CCR-9984553.

[1] DME-based zero-skew algorithms can be adapted to handle the prescribed-skew problem, where a legitimate skew schedule that defines the exact relative skews among clock pins is given (see [17]). In fact, this is a special case of our UST/DME algorithms.

When a skew is excessively *positive* or *negative*, the data is produced too late to be clocked in (zero-clocking) or too early that it races through to the next flip-flip (double-clocking), respectively. To avoid any incorrect operation, we must bound the skew from above and below 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. To correctly latch in data, the amount of time that the data has 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 or consume less power [16]. For example, we can make a circuit more robust to process variations by adding a *safety margin* $\delta_l (\geq 0)$ to the lower bound constraint in Eqn. (1), or subtracting a safety margin $\delta_u (\geq 0)$ from the upper bound constraint in Eqn. (2):

$$t_i - t_j \;\geq\; t_{hold,\max} - t_{pFF,\min} - t_{logic,\min} + \delta_l, \quad (3)$$
$$t_i - t_j \;\leq\; C_P - t_{pFF,\max} - t_{logic,\max} - t_{setup,\max} - \delta_u. \quad (4)$$

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 $\mathcal{C} = \{t_i - t_j \in R_{i,j}\}$ to denote the set of skew constraints for all sequentially adjacent clock pins s_i and $s_j \in \mathcal{S}$. 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 $\mathcal{C}$, i.e., $l_{i,j} \leq skew_{i,j} \leq u_{i,j}, \forall s_i, s_j \in \mathcal{S}$.

The problem we address in this paper can be stated as follows: **Minimum-Length Useful Skew Routing Tree Problem:** Given the clock pin locations of $\mathcal{S} = \{s_1, ..., s_n\}$ and a set of skew constraints $\mathcal{C} = \{t_i - t_j \in R_{i,j} = [l_{i,j}, u_{i,j}]\}$, for clock pins $s_i, s_j \in \mathcal{S}$, find a clock tree T connecting $\mathcal{S}$ such that the total length is minimized subject to skew constraints $\mathcal{C}$.

3. SKEW SCHEDULING

We use a constraint graph $G_C = (V, E)$ to capture skew constraints as in [3; 4]. 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 by $w_{i,j}$, is $-l_{i,j}$, and the weight of $e_{j,i}$, denoted by $w_{j,i}$, is $u_{i,j}$. Consider a 3-pin example with the following local skew constraints:

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

Figure 1(a) is the corresponding constraint graph.

Note that even though zero skew is within $R_{1,2}$, setting $skew_{1,2}$ equal to zero would render the other two skew constraints $t_1 - t_3 \in R_{1,3}$ and $t_2 - t_3 \in R_{2,3}$ not simultaneously satisfiable. A careful study, to be 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). Although a similar concept called *permissible skew range* was used in [9], our work in this paper is the first to use a graph-based approach to compute FSR's, as well as to use them for incremental skew scheduling and clock routing simultaneously.

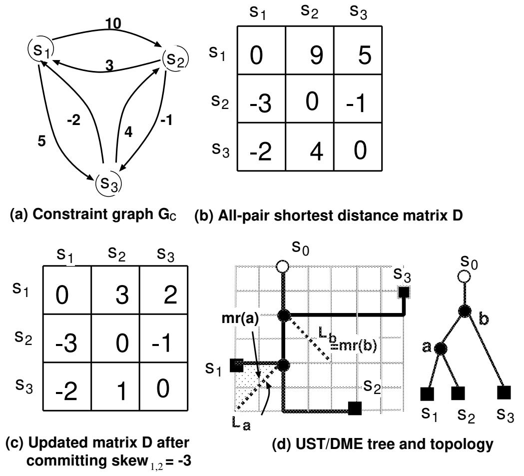


Figure 1: An example showing the incremental skew scheduling based on an all-pair shortest-distance matrix. (a) The constraint graph G_C capturing the local skew constraints: (i) $t_1 - t_2 \in [-10, 3] = R_{1,2}$, (ii) $t_1 - t_3 \in [-5, -2] = R_{1,3}$, and (iii) $t_2 - t_3 \in [1, 4] = R_{1,3}$. (b) The all-pair shortest distance matrix $\mathcal{D}$ constructed from G_C. (c) The updated matrix after committing the skew of s_1 and s_2 to $skew_{1,2} = -3$. (d) The clock layout (left) and its corresponding abstract topology (right). The shaded region is the merging region $mr(a)$ constructed by merging s_1 and s_2 under the linear delay model. Merging region $mr(b)$ is constructed by merging s_3 and a (using segment L_a from $mr(a)$).

3.1 Computation of Feasible Skew Ranges

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 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:

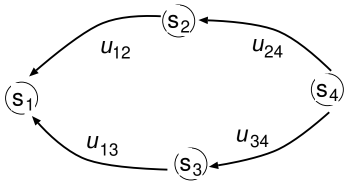
$$(t_1 - t_2) + (t_2 - t_4) \;\leq\; u_{1,2} + u_{2,4},$$
$$(t_1 - t_3) + (t_3 - t_4) \;\leq\; u_{1,3} + u_{3,4}.$$

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

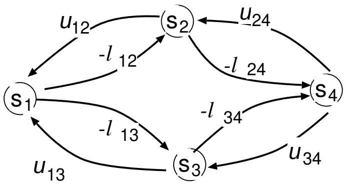
$$t_1 - t_4 \leq \min\{u_{1,2} + u_{2,4}, u_{1,3} + u_{3,4}\}.$$

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 2(a). Therefore, the upper bound of the implied skew constraint between s_1 and s_4 is equal to the shortest distance from s_4 to s_1.

Next, we consider the four lower-bound local skew constraints. As $l_{i,j} \leq t_i - t_j$ 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 (see Figure 2(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



(a) constraint graph for upper bounds



(b) constraint graph for upper/lower bounds

Figure 2: (a) A constraint graph constructed from four upper-bound skew constraints: $t_1 - t_2 \leq u_{1,2}$; $t_2 - t_4 \leq u_{2,4}$; $t_1 - t_3 \leq u_{1,3}$; and $t_3 - t_4 \leq u_{3,4}$. (b) The corresponding lower-bound skew constraints are added: $l_{1,2} \leq t_1 - t_2$; $l_{2,4} \leq t_2 - t_4$; $l_{1,3} \leq t_1 - t_3$; and $l_{3,4} \leq t_3 - t_4$.

from s_1 to s_4. When G_C contains no negative cycles, the shortest distance from s_i to s_j, denoted by $d_{i,j}$, is well-defined. It is proved in [3] that feasible skew schedules subject to $\mathcal{C}$ exist if and only if G_C contains no negative cycles.[2]

Therefore, given a constraint graph G_C without negative cycles, we can build an all-pair 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 [3] to represent all FSR's. (If the graph is sparse, a faster algorithm can be employed [3].) The maximum feasible skew range between s_i and s_j, under all the implicit and explicit constraints, can be simply expressed as $FSR_{i,j} = [-d_{i,j}, d_{j,i}]$. Hence, we also refer to $\mathcal{D}$ as the FSR matrix. Figure 1(b) is the resultant FSR matrix for the original constraint graph in (a). In general, for a bounded-skew routing with an absolute 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. In the example given in Figure 1, no feasible absolute global skew bound exists.

3.2 Incremental Scheduling

In the following, we informally show that if we start with a constraint graph G_C without negative cycles, then we do not create any negative cycle 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, so do all FSR's. 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 while a feasible skew schedule is always guaranteed.

Suppose G_C has no negative cycles. For simplicity, we assume that G_C is a complete graph; if s_i and s_j are not sequentially adjacent, then we set the edge weights $w_{i,j}$ and $w_{j,i}$ equal to ∞. Consider narrowing $FSR_{i,j} = [-d_{i,j}, d_{j,i}]$ to a subrange $[x, y]$ where $-d_{i,j} \leq x \leq y \leq d_{j,i}$. We decrease the weights of $e_{i,j}$ and $e_{j,i}$ in G_C from $w_{i,j} \geq -d_{i,j}$ to $w_{i,j} = -x$ and from $w_{j,i} \geq d_{j,i}$ to $w_{j,i} = y$, respectively. We make the following two observations:

1. The new shortest distances from s_i to s_j and from s_j to s_i are respectively $-x$ and 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 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 two observations, we conclude that if some negative cycle is created, the two-node cycle linking s_i and s_j must be a cycle with a negative weight of $-x + y < 0$. However, this contradicts the fact that $x \leq y$ (i.e., $[x, y]$ is non-empty).

Therefore, we have the following theorem and corollary:

THEOREM 1. *Suppose the constraint graph G_C constructed from skew constraints $\mathcal{C}$ has no negative cycles. Then, shrinking $FSR_{i,j} = [-d_{i,j}, d_{j,i}]$ to a subrange $[x, y]$ such that $-d_{i,j} \leq x \leq y \leq d_{j,i}$ does not create any negative cycle in G_C. Therefore, all entries of the FSR matrix $\mathcal{D}$ are well-defined, and the matrix can be updated as follows:*

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

Equivalently, any $FSR_{k,l}$ for $1 \leq k \neq l \leq n$ exists. □

COROLLARY 1. *Suppose a feasible skew schedule subject to skew constraints $\mathcal{C}$ exists. Then, a feasible skew schedule subject to tighter skew constraints $\mathcal{C}'$ by shrinking some feasible skew range $FSR_{i,j} = [-d_{i,j}, d_{j,i}]$ to a subrange $[x, y]$ such that $-d_{i,j} \leq x \leq y \leq d_{j,i}$ still exists.* □

Input: Skew commitment $skew_{i,j} = x$; An FSR matrix $\mathcal{D} = \{d_{i,j}\}$
Output: An updated matrix $\mathcal{D}$
Set $d_{i,j} \leftarrow -x$ and $d_{j,i} \leftarrow x$ **for** each $d_{k,l}, 1 \leq k \neq l \leq n$ in $\mathcal{D}$ Set $d_{k,l} \leftarrow \min\{d_{k,l}, d_{k,i} - x + d_{j,l}, d_{k,j} + x + d_{i,l}\}$

Figure 3: Procedure for updating the FSR matrix $\mathcal{D}$ after a skew commitment.

When we narrow a non-trivial FSR to a single skew value, a *skew commitment* is made. Figure 3 shows an $O(n^2)$ algorithm to update the FSR matrix $\mathcal{D}$ 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 1(c) shows the resultant matrix $\mathcal{D}$ after we commit to $skew_{1,2} = -3$: $FSR_{1,3} = [-2, -2]$ and $FSR_{2,3} = [1, 1]$.

We now construct an $O(n^3)$ algorithm for incremental scheduling. First, an all-pair shortest path algorithm is used to generate the FSR matrix $\mathcal{D}$. While there exist uncommitted skews, we select one of them and commit it to a value within the corresponding FSR. 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 determine skew commitments in a greedy manner such that the clock tree wirelength is minimized.

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 DME-based paradigm:

[2]In [3], a feasible skew schedule (if 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 $\mathcal{C}$.

1. A bottom-up phase to construct a binary tree of merging regions (or segments) that represent the loci of possible embedding points of internal nodes, and

2. A top-down phase to determine the exact locations of the internal nodes.

As the top-down phase is identical to those presented in [1; 2], we emphasize only the bottom-up phase in this paper.

In the UST/DME paradigm, FSR's play an important role in determining the construction of merging regions. This is significantly different from previous approaches where skew scheduling plays a very passive role in clock layout synthesis.

Here, we distinguish a physical interconnect tree T from its *abstract* topology $\mathcal{T}$. An abstract topology is a binary tree (with the possible exception at the root node, denoted by s_0) such that all clock pins are the leaf nodes of the binary tree. T is an embedding of the abstract topology $\mathcal{T}$, i.e., each internal node $v \in \mathcal{T}$ is mapped to a location $l(v) = (x_v, y_v)$ in the Manhattan plane, where (x_v, y_v) are the x- and y-coordinates. Figure 1(d) shows a physical layout (on the left) and its abstract topology (on the right).

The prescribed topology that UST/DME takes in is essentially an abstract topology. We use T_v to denote the physical tree rooted at node v. An abstract topology gives the order of merging in the physical tree: if v is the parent node of u and w in $\mathcal{T}$, then T_u and T_w are first constructed before they are merged at v to form T_v. In Figure 1(d), 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 source node directly. In contrast, Greedy-UST/DME constructs the abstract topology from the physical layout process.

4.1 The UST/DME Algorithm

The outline of UST/DME is given in Figure 4. At the beginning of the algorithm, we have a forest of singleton trees T_{s_i}'s, each containing a pin in $\mathcal{S}$. The order of subtree merging is determined by a topological sort (in a bottom-up fashion) of the prescribed topology $\mathcal{T}$. To determine the skew value for wirelength minimization at each DME merging step, we need to compute and store the FSR's 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-pair shortest distance matrix $\mathcal{D} = \{d_{i,j} : s_i, s_j \in \mathcal{S}\}$ from G_C in $O(n^3)$ to represent all FSR's. While the forest has more than one tree, 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 determine the merging region of v, denoted by $mr(v)$, based on $mr(u)$ and $mr(w)$, the merging regions of v's children. The computation of the parent merging region from two child merging regions is presented next.

4.2 Merging Region Construction

In the BST/DME approach, the absolute global skew bound B (≥ 0) 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. Such a simplification greatly facilitates the computation of merging regions 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* [2]. They can either be a *Manhattan arc* (a segment with slope 1 or -1) 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

Input: 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

1. Construct constraint graph $G_C = (V, E)$ from $\mathcal{C}$
2. If G_C has negative cycles, return no solution
3. Build an all-pair 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 $\mathcal{T}$
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 on L_u
9. Update $\mathcal{D}$ for skew commitment $skew_{i,j} = x$ (Figure 3)
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 and $mr(u) = L_u$
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 on $L_v \in mr(v)$ in top-down order)

Figure 4: The UST/DME algorithm.

such skew properties, the feasible skew range between clock pins in T_u and T_w can be easily computed under the absolute global skew bound constraint 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, then merging region $mr(v)$ is constructed within $SDR(L_u, L_w)$ and all skews between clock pins in T_u and T_w are within $[-B, B]$.

However, the computation of merging regions under incremental scheduling in our UST/DME approaches introduces a problem not encountered in BST/DME. 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 children's merging regions is non-trivial. Selection of a joining segment L_u from $mr(u)$ implies commitments to some skew values for clock pins in T_u. Those commitments affect the FSR's 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's at different location on L_u will have different impacts on the feasible skew ranges for clock pins in the other subtree. This further complicates the merging regions construction under various local skew constraints. Despite these difficulties, UST/DME 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 UST/DME to only Manhattan arcs. 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 $x \in FSR_{i,j}$ [2]. Let L_u be 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. As a skew commitment of $skew_{i,j} = x$ changes $FSR_{k,l}$ for clock pins s_k and s_l in T_w, we re-compute 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 re-computed $mr(w)$ such that it is closest to L_u. Finally, $mr(v)$ is then constructed from L_u and L_w such that the skew between $s_i \in T_u$ and $s_k \in T_w$ is feasible, i.e., $skew_{i,k} \in FSR_{i,k}$.

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 parent merging region $mr(v)$.

Figure 1(d) shows the resulting tree routing under the linear delay model after we commit $skew_{1,2} = -3$ and hence, $skew_{1,3} = -2$ and $skew_{2,3} = 1$. The resulting feasible skew schedule is $\{t_1 = 6, t_2 = 9, t_3 = 8\}$. The shaded region is the merging region $mr(a)$ constructed by merging T_{s_1} and T_{s_2}, using s_1 and s_2 as the joining segments. Merging region $mr(b)$ is constructed by merging T_{s_3} and T_a, using joining segments s_3 and L_a in $mr(a)$.

4.3 The Greedy-UST/DME Algorithm

When no prescribed topology is given, Greedy-UST/DME can be applied to determine the topology of the tree of merging regions (or joining segments) in a greedy bottom-up fashion as in Greedy-DME [6]. The outline of Greedy-UST/DME is similar to that of UST/DME except for step 4, which is amended as below:

4. **for** each merging of subtrees T_u and T_w to form T_v based on the nearest neighbor graph [6]

We have a forest $\mathcal{F}$ of singleton trees T_{s_i}'s at the beginning of the algorithm. The order of subtree merging is determined from the so-called nearest neighbor graph (NNG) [6], which stores the promising merging pairs of subtrees based on the estimated merging cost. The algorithm first constructs an NNG that maintains the nearest neighbor (in terms of merging cost) of all merging regions of root nodes of subtrees 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 non-decreasing order of merging cost for merging, where k is a constant ranging from 2 to 5 [6]. The NNG is re-constructed after each iteration. The number of subtrees is reduced to $(1-1/k)|\mathcal{F}|$ after each iteration. The algorithm finishes in $\log_{k/(k-1)} n$ iterations, where n is the number of sinks.

Similar to the approaches in [5; 6], Greedy-UST/DME merges two subtrees at their root nodes. In contrast, BST/DME allows merging at non-root nodes by dynamically changing subtree topologies before the merging is performed. The feature of dynamically changing subtree topologies enables BST/DME to produce very low-cost solutions when the skew bound is large. To allow such a feature in Greedy-UST/DME, we would have to change all the previously committed skew values in these subtrees when their topologies change. However, when a skew commitment is changed (or in general, when an FSR is updated to a range that is outside its original range), the incremental update algorithm (Figure 3) is no longer applicable, and the FSR matrix $\mathcal{D}$ has to be rebuilt by applying the Floyd-Warshall algorithm [3]. The time complexity of the scheduler in Greedy-UST/DME would then be at least one order higher. To avoid that, 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 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 has the same linear time complexity like other DME-based approaches [5; 6]. At each of the $n-1$ merging steps in the bottom-up merging phase, UST/DME makes a skew commitment and updates the FSR matrix $\mathcal{D}$ accordingly in $O(n^2)$. Therefore, UST/DME has an overall complexity of $O(n^3)$.

The complexity of Greedy-UST/DME is determined by the complexities of Greedy-DME and the incremental scheduler. As the most efficient implementation of Greedy-DME using the bucket decomposition method of [6] has asymptotic linear time complexity, the overall complexity of Greedy-UST/DME is solely determined by the incremental scheduler. Hence, the complexity of Greedy-UST/DME is $O(n^3)$.

5. EXPERIMENTS

Table 1: The pin counts, the distribution of skew constraints, and the maximum allowable absolute global skew bounds of the benchmark circuits from [17].

circuit	number of clock pins	number of skew constraints	maximum absolute global skew bound (ns)
s1423	74	78	1.4
s5378	179	175	0.3
s15850	597	318	0.2

The proposed UST/DME and Greedy-UST/DME algorithms have been implemented in C++ and tested on three ISCAS89 benchmark circuits from [17] on a Sun UltraSPARC-II. Table 1 gives the pin counts, the distribution of skew constraints, and the maximum absolute global skew bounds of the three benchmark circuits.

The Elmore delay model is used to measure the wire delays. We set the k parameter for Greedy-UST/DME to 2, i.e., in each iteration, $\mathcal{F}/2$ independent nearest-neighbor pairs in the NNG are merged. We compare both UST/DME and Greedy-UST/DME with BST/DME [2] and UST-BP [17]. For UST-BP, we report only the wirelength reduction over the zero-skew routing but not the actual wirelength because of a discrepancy between wirelength units. The results are summarized in Table 2. For UST/DME, we use topologies generated by BST/DME [2].

Greedy-UST/DME performs 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 as they share the same topologies, and both of them outperform Greedy-UST/DME and UST-BP significantly. As 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 1. As a result, the BST/DME and UST/DME solutions are very close to minimal Steiner trees due to the large skew bounds.

We also compare Greedy-UST/DME and UST/DME with the BST/DME algorithm, using the largest benchmark s15850 under different safety margins $\delta = \delta_l = \delta_u$ (see Eqns. (3) and (4)) ranging from 0 to 250 pico-seconds (ps). As the UST-BP code is unavailable, we cannot make a similar comparison. For each safety margin $\delta \in [0, 200]$ps, the absolute global skew bound for BST/DME is $B = (200 - \delta)$ps, obtained by intersecting all feasible skew ranges. BST/DME is infeasible when the safety margin δ is beyond 200ps. For larger safety margins ($\geq$ 190ps), which are more desirable for high-performance circuits, Greedy-UST/DME is obviously superior to BST/DME. For smaller safety margins,

Table 2: Comparison of wirelengths among BST/DME [2], UST-BP [17], Greedy-UST/DME, and UST/DME (with BST/DME topology) in terms of wirelength reduction over zero-skew trees obtained by the DME approach [5].

circuit	ZST wire	BST/DME wire	BST/DME reduction over ZST	UST-BP reduction over ZST	Greedy-UST/DME wire	Greedy-UST/DME reduction over ZST	Greedy-UST/DME CPU time (min:sec)	UST/DME (with BST/DME topology) wire	UST/DME (with BST/DME topology) reduction over ZST	UST/DME (with BST/DME topology) CPU time (min:sec)
s1423	107278	65207	39%	23%	90118	16%	0:14	66106	38%	0:01
s5378	176518	107527	39%	21%	149292	15%	1:09	108219	38%	0:08
s15850	448599	274901	39%	16%	355561	20%	9:23	277281	38%	4:48

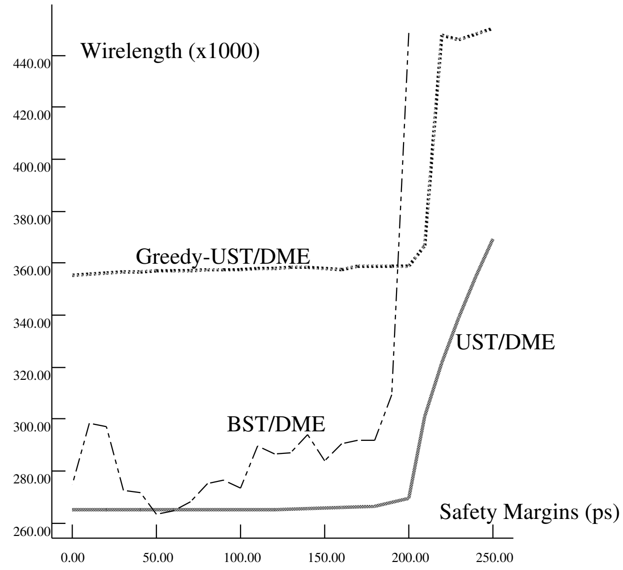


Figure 5: Comparisons of total wirelength among BST/DME, Greedy-UST/DME, and UST/DME (with a BST/DME topology) on ISCAS89 s15850 test case for different safety margins.

however, it is inferior to BST/DME because the corresponding skew bounds for BST/DME are relatively large. Moreover, Greedy-UST/DME solutions are as good as, if not better than, the zero-skew solution for any safety margin. Furthermore, we run UST/DME using an abstract topology obtained by BST/DME. It outperforms BST/DME because of its simultaneous skew scheduling and tree routing.

Finally, note that the wirelengths of UST/DME solutions increase rapidly when the safety margin δ approaches 250ps. This is due to the considerable amount of detour wire used to achieve very large skew value for some merging of subtrees. Instead of using long detour wire, a more practical approach for achieving merging with a large skew value is to use repeaters. We are currently working along this direction in order to make our method more suitable for practical clock designs.

6. CONCLUSIONS

In this paper, 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. We are seeking to further improve our incremental skew scheduler that allows more flexible adjustments of subtree topologies before the merging of subtrees. We are also trying to reduce the complexity $O(n^3)$ of the skew scheduler by using the property of sparse constraint graph. Furthermore, we are integrating the UST/DME approaches with repeater insertion for large practical industry-scale clock designs.

7. REFERENCES

[1] T.-H. Chao, Y.-C. Hsu Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng. Zero skew clock routing with minimum wirelength. *IEEE Trans. on Circuits and Systems*, 39(11):799–814, November 1992.

[2] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao. Bounded-skew clock and Steiner routing. *ACM Trans. on Design Automation of Electronics Systems*, 3(3):341–388, 1998.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*, chapter 25.5, pages 539–543. 1990.

[4] R. B. Deokar and S. S. Sapatnekar. A graph-theoretic approach to clock skew optimization. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 407–410, 1994.

[5] M. Edahiro. Minimum path-length equi-distant routing. In *Proc. IEEE Asia-Pacific Conf. on Circuits and Systems*, pages 41–46, December 1992.

[6] M. Edahiro. An efficient zero-skew routing algorithm. In *Proc. Design Automation Conf*, pages 375–380, June 1994.

[7] J. P. Fishburn. Clock skew optimization. *IEEE Trans. on Computers*, 39(7):945–951, July 1990.

[8] A. B. Kahng and C.-W. A. Tsao. Practical bounded-skew clock routing. *Journal of VLSI Signal Processing (Special issue on High Performance Clock Distribution Networks)*, 16(2-3):199–215, June-July 1997.

[9] J. L. Neves and E. G. Friedman. Optimal clock skew scheduling tolerant to process variations. In *Proc. Design Automation Conf*, pages 623–628, 1996.

[10] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun. checkTc and minTc: Timing verification and optimal clocking of synchronous digital circuits. In *Proc. Int. Conf. on Computer Aided Design*, pages 552–555, 1990.

[11] N. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Graph algorithms for clock schedule optimization. In *Proc. Int. Conf. on Computer Aided Design*, pages 132–136, 1992.

[12] T.G. Szymanski. Computing optimal clock schedules. In *Proc. Design Automation Conf.*, pages 399–404, 1992.

[13] R.-S. Tsay. An exact zero-skew clock routing algorithm. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, CAD-12(2):242–249, February 1993.

[14] A. Vittal, H. Ha, F. Brewer, and M. Marek-Sadowska. Clock skew optimization for ground bounce control. In *Proc. Int'l Conf. on Computer Aided Design*, pages 395–399, Nov. 1996.

[15] P. Vuillod, L. Benini, A. Bogliolo, and G. De Micheli. Clock-skew optimization for peak current reduction. In *Proc. Int'l Symp. on Low Power Electronics and Design*, pages 265–270, Aug. 1996.

[16] J. G. Xi and W. W.-M. Dai. Jitter-tolerant clock routing in two-phase synchronous systems. In *Proc. Int. Conf. on Com-*

puter Aided Design, pages 316–320, 1996.

[17] J. G. Xi and W. W.-M. Dai. Useful-skew clock routing with gate sizing for low power design. *Journal of VLSI Signal Processing Systems*, 16(2/3):163–170, 1997.