

Associative Skew Clock Routing for Difficult Instances *

Min-Seok Kim

Department of Electrical and Computer Engineering
Texas A&M University
College Station, TX 77843-3128, USA

Jiang Hu

Department of Electrical and Computer Engineering
Texas A&M University
College Station, TX 77843-3128, USA
jianghu@ece.tamu.edu

Abstract

In clock network synthesis, sometimes skew constraints are required only within certain groups of clock sinks and do not exist between different groups. This is the so-called associative skew clock routing problem. Although the number of constraints is reduced, the problem becomes more difficult to solve due to the enlarged solution space. The perhaps only previous work used a very primitive delay model and cannot handle difficult instances in which sink groups are intermingled. We reuse existing techniques to solve this problem, including the difficult instances, based on a more accurate and popular delay model. Experimental results show that our algorithm can reduce the total clock routing wirelength by 12% on average compared to greedy-DME which is one of the best zero skew routing algorithms.

1. Introduction

Clock network is of paramount importance to both performance and power efficiency of integrated circuits. A common goal of clock network synthesis is to minimize clock network size subject to skew constraints. Skew is defined as the difference of clock signal arrival times at sinks (or flip-flops). A small clock network usually implies less power dissipation and power supply noise. Skew constraints ensure that circuits operate properly at desired frequency. In general, skew constraint may take one of the following forms:

- *Zero skew* [1, 5, 13]. This requires that clock signal arrival times are the same for all sinks. Zero skew constraint is very popular in industry due to its simplicity.
- *Bounded skew*. In this case, a skew does not have to be zero and can be any value within a bound. The bound can be either a global one for all sinks [4] or a local one for each pair of clock sinks [12]. The relaxed skew constraints normally lead to smaller clock network size compared to zero skew routing.
- *Prescribed skew* [2, 14]. The skew for each pair of sinks is required to satisfy a usually non-zero target. This is for the purpose of improving circuit operating frequency [6, 9], reducing power supply noise [8] or improving tolerance to variations [7].

Another form of skew constraint, *associative skew*, has been rarely mentioned in literature before. In an associative skew clock routing problem [3], the clock sinks are partitioned into a few groups. Skew constraints are required only within each group and do not exist between different groups. This problem is more difficult to solve than conventional zero/prescribed skew routings because the reduction of constraints results in increased solution space to be searched.

To the best of our knowledge, [3] is the only previous work attempting to solve this problem. Although several heuristic solutions were proposed in [3], all of them are based on a very primitive delay model which equalizes clock signal delay with geometrical path length. Moreover, they are effective only when the sink groups are geometrically separated from each other. In the difficult but common instances where the sink groups are intermingled, the algorithms developed in [3] perform worse than a simple exten-

* This work is partially supported by Semiconductor Research Corporation under contract 2004-TJ-1205.

sion to greedy-DME [5], which is one of the best zero skew routing algorithms.

Why is there so few work on this problem? Perhaps people have a simple solution in practice. That is, enforce arbitrary or empirical skew constraints between each pair of groups so that this problem becomes a conventional clock routing problem. For example, one may require the skew among all groups to be zero if intra-group skew constraint is zero for all groups. In fact, this is the extension made by [3] to greedy-DME [5] in order to compare their algorithms. However, it is not obvious how to find the best inter-group constraints for minimizing clock routing wirelength. Our experiments show that an elaborated method can reduce wirelength by about 12% compared to the naive approach of adding zero skew constraints. Such amount of reduction may be unimportant for old technologies, but would provide precious power savings in today’s power-hungry IC designs [11]. Therefore, an elaborated solution to the associative skew routing problem becomes increasingly necessary, especially for the aforementioned difficult instances. It is stated in [3]: “the key open issue is to find a heuristic that consistently outperforms greedy-DME for the domain with intermingled sink groups.”

In this work, we will show that the associative skew routing problem can be solved by carefully modifying existing bounded skew routing algorithm [4] instead of seeking completely new techniques as in [3]. Experimental results on benchmark circuits with the difficult instances show that our approach can reduce clock routing wirelength by 12% on average compared to the extended greedy-DME method. In other words, we solved the key open issue raised in [3].

2. Problem Formulation

In a clock tree, if clock signal delays to sink (flip-flop) a and b are t_a and t_b , respectively, the skew between them is defined as $t_a - t_b$. Given a set of clock sinks which are partitioned into k groups G_1, G_2, \dots, G_k , each pair of sinks $s_{i,a}$ and $s_{i,b}$ in the same group G_i have a certain skew constraint but there is no skew constraint between any two sinks $s_{i,a}$ and $s_{j,b}$ which belong to two different groups G_i and G_j . For the simplicity of discussions, we let intra-group skew constraints be zero, i.e., clock signal delay to every sink in the same group has to be the same. Our method can be extended to non-zero prescribed skew or bounded skew constraint easily. The associative skew tree (AST) routing problem is to construct a clock tree such that the total wirelength is minimized while the intra-group skew constraints are satisfied.

Compared to conventional zero/prescribed skew routing, the formulation of AST implicitly leads to a by-product in addition to the clock tree itself. That is the skew among different sink groups which is not available in the input. The

skew between group G_i and G_j is denoted as $S_{i,j}$. In [3], this inter-group skew is called offset. We need to specify the inter-group skew $S_{i,j}$ for all groups either implicitly or explicitly in solving the AST problem.

3. Delay Model

In contrast to the previous work [3] which uses path length based metric, we employ the Elmore delay model as in many other clock routing works [1, 2, 4, 5, 12, 13]. Although the Elmore delay is often inaccurate, it works very well for clock tree routing. One major reason of the inaccuracy is its neglect of the resistive shielding effect [10]. This is particularly significant when estimating the delay of a node near the source. However, the delay estimations in a clock tree are mainly for those far-end leaf nodes. Moreover, the error of *skew* estimation is usually very small despite large errors on *delay* estimation. In other words, the error in delay estimation is largely cancelled out in the subtraction operation when calculating skew. We have observed this phenomenon when we compare the Elmore based skew with SPICE simulation results.

4. Observation

Why is the previous work [3] incapable of handling the instances where sink groups are intermingled? This is because that [3] constructs subtrees for each sink group separately and then stitches them together. Such approach may result in wire overlaps between subtrees of different groups despite the sophisticated stitching techniques [3]. In general, wire overlap implies inefficiency on wire usage. In Figure 1(a), two dark (light) sinks from the same group are merged at node j (k). We use T_j to represent a subtree rooted at node j . Then, subtree T_j and T_k are merged at the source node. The intermingling among different sink groups implies strong proximity connections among them. Therefore, constructing trees separately for each group would conflict with such strong connections. If we allow sinks from different groups (with different grayscale) to be merged as in Figure 1(b), the wirelength can be remarkably reduced.

5. Algorithm

5.1. Overview and Merging Order

The observation of previous section tells that we should handle sinks from all groups simultaneously instead of separately in AST routing. This makes our AST construction more like traditional approaches [1, 2, 4, 5, 12, 13] where the clock tree is constructed by successively merging subtrees in a bottom-up manner. Initially, each subtree is a sin-

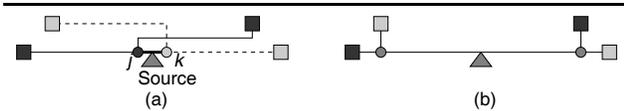


Figure 1. Sink group is indicated by grayscale of sinks. Constructing trees for each group separately and then stitching them together as in (a) may result in wire overlap and large wirelength. Allowing mergings between sinks from different groups as in (b) may reduce wirelength.

gle sink. Two subtrees can be merged to form a new subtree. The merging is repeated till only one subtree is left and this single subtree is connected to the clock source directly. Please note that this clock tree routing procedure is independent of the location of clock source.

The merging order in our approach is the same as the nearest neighbor method in the greedy-DME algorithm [5]. That is, we always first merge the pair of subtrees with the minimum distance between their roots. When looking for the min-distance pair, instead of comparing the distance of all pairs, a nearest neighbor graph, which is a planar graph, can be constructed to speed-up the search. In [5], the nearest neighbor graph is built based on Delaunay triangulation. We implement it according to a spanning graph [15] which is better for Manhattan space. Please note that we allow sinks from different groups to be merged. This is the key difference from the work of [3].

5.2. Layout Embedding

When merging two subtrees, we need to determine the location of the root (merging node) of the new subtree formed from the merging. The location of the root needs to ensure that skews among sinks in the new subtree satisfy skew constraints. For the Elmore delay based clock routings, such location can be found by solving an equation that matches the delay of two subtrees [13]. The procedure of finding the location of merging nodes is called layout embedding.

A famous layout embedding technique is **DME** (Deferred Merge Embedding) [1]. Instead of being fixed at a single location, a merging node can slide along a segment (along $\pm 45^\circ$ direction) without violating skew constraints. Such segment is called *merging segment* which is illustrated by the dashed lines in Figure 2(a). If the skew constraint is a bounded range instead of a single value, the location of a merging node can be anywhere in a specific polygon region to satisfy the skew bound. This region is called *merging region* in **BST** (Bounded Skew Tree) routing [4]. A merging

region can be treated as a set of parallel merging segments. In Figure 2(b), merging regions are indicated by shaded area. In this paper, the merging segment and merging region corresponding node i are denoted as MS_i and MR_i , respectively.

After the bottom-up merging procedure, another top-down traversal of the tree is performed to decide the exact merging locations on each merging segment such that the total wirelength is minimized.

5.3. Merging Subtrees from the Same Group

If all sinks of a subtree T_a belong to a same group G_j , we say that this subtree is from group G_j . When merging two subtrees from the same group (Figure 2(a)), the scenario is almost the same as the classic DME embedding [1]. The fact that two subtrees T_a and T_b are from the same group is denoted as $T_a \bowtie T_b$. Since there is a skew constraint between two subtrees from the same group, we can use the DME technique to find a single merging segment MS_c for them as shown in Figure 2(a).

5.4. Merging Subtrees from Different Groups

When merging two subtrees from different groups, the scenario is very similar to the BST routing [4]. Using Figure 2(b) as an example, T_a and T_b are from group G_1 and group G_2 , respectively. Since there is originally no skew constraint between G_1 and G_2 , there is no restriction for the location of their merging node. However, we prefer the wirelength from this merging to be the minimal (which equals the Manhattan distance between node a and b). Thus, the location of the merging is restricted to a merging region [4] (shaded in Figure 2(b)) between merging segment MS_a and MS_b . In BST [4], the merging region is constructed in a way such that the skew bound is satisfied. Since there is no skew bound for different sink groups in AST, the merging region is chosen as the *shortest distance region (SDR)* [4] between them. Please note that this merging region implies a bounded range for skew $S_{2,1}$ between group G_1 and G_2 . If two subtrees are from different groups and their root nodes are represented by merging *regions* instead of merging *segments*, the merging is the same as that in BST.

5.5. Merging Subtrees from Partially Shared Groups

The scenario of merging subtrees from partially shared groups does not exist in either DME [1] or BST [4]. Besides, it is more complicated than the scenarios in previous two subsections. We will use the examples in Figure 2 to illustrate a few instances in this scenario.

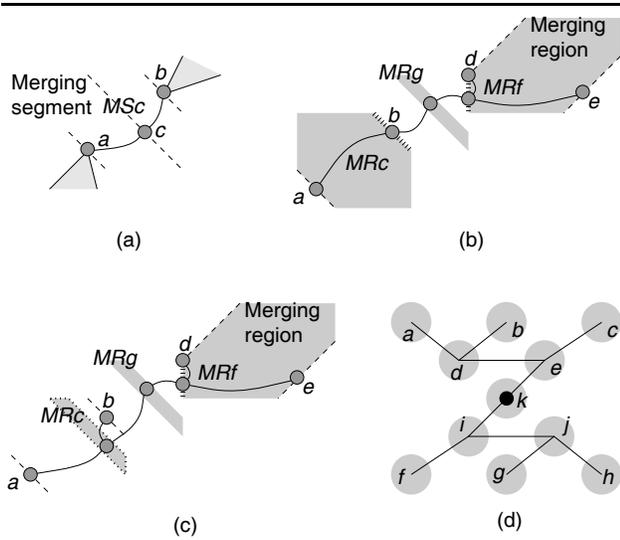


Figure 2. (a) Merging subtree T_a and T_b from the same group at merging segment MR_c for merging node c with DME [1]. (b) If two subtrees belong to different groups, a merging region (shaded) is formed between them like in BST [4]. If T_a and T_d are from the same group ($T_a \bowtie T_d$), a reduced merging region MR_g is formed when merging T_c and T_f . (c) If $T_a \bowtie T_d$ and $T_b \bowtie T_e$, MR_c is reduced to the polygon with dotted boundary. (d) A shaded circle represents a merging region. If $T_a \bowtie T_g$, $T_b \bowtie T_h$ and $T_c \bowtie T_f$, the merging regions are reduced according to intersection parts of the bounded skew range.

Instance 1: Consider the example in Figure 2(b) where subtree T_a and T_d are from group G_1 , T_b is from group G_2 and T_e is from group G_3 . Therefore, $T_a \bowtie T_d$. First, T_a and T_b are merged to form T_c , and T_d and T_e are merged to form T_f .

Now we are trying to merge T_c and T_f . Similar as BST [4], we choose the northeast boundary segment of merging region MR_c and the west boundary segment of merging region MR_f for this merging. In Figure 2(b), these two boundary segments are dotted. The reason to choose them is that they are the closest boundaries between MR_c and MR_f .

The northeast boundary of MR_c corresponds to one merging segment in MR_c . Since the merging segment implies a unique skew between T_a and T_b , a skew constraint $S_{2,1}$ between G_2 and G_1 is induced. The west boundary of MR_f corresponds to the southwest end points of a subset of merging segments in MR_f . Since each merging segment implies a skew between T_d and T_e , a subset of

merging segment implies a bounded range for skew $S_{3,1}$ between group G_3 and G_1 .

As both T_c and T_f contain sinks from group G_1 , the merging between them has to satisfy the skew constraint within G_1 . According to this skew constraint, we can find the merging region MR_g for this merging as shown in Figure 2(b). After this merging, G_1 , G_2 and G_3 can be treated to form a new group $G_{1,2,3} = G_1 \cup G_2 \cup G_3$ and there are prescribed/bounded skew constraints within this new group.

Instance 2: In Figure 2(c), both T_a and T_d are from group G_1 , and T_b and T_e are from group G_2 , i.e., $T_a \bowtie T_d$ and $T_b \bowtie T_e$. Similar as BST [4], we choose the west boundary of MR_f for the merging between T_c and T_f . This boundary determines that the skew $S_{2,1}$ between G_2 and G_1 has to be in a bounded range. Consequently, we have to shrink the merging region MR_c such that the skew between T_a and T_b satisfies the bound for $S_{2,1}$. The rest steps of the merging is the same as that in BST.

Instance 3: In Figure 2(d), $T_a \bowtie T_g$, $T_b \bowtie T_h$ and $T_c \bowtie T_f$. Now we consider the merging at node k . The merging region MR_d implies a skew bound $S_{a,b}$ between T_a and T_b . Similarly, the merging region MR_j implies another skew bound $S_{g,h}$ between T_g and T_h . If both T_a and T_g are from group G_1 , and both T_b and T_h are from group G_2 , the skew $S_{2,1}$ between group G_2 and G_1 has to satisfy both $S_{a,b}$ and $S_{g,h}$. In other words, The skew bound for $S_{2,1}$ should be the intersection part between $S_{a,b}$ and $S_{g,h}$. For example, if $S_{a,b} \in [-3, 8]$ and $S_{g,h} \in [-7, 4]$, then $S_{2,1} \in [-3, 4]$. If the intersection part is empty, wire snaking [13] is induced. If both T_c and T_f are from group G_3 , the skew bounds for $S_{3,2}$ and $S_{3,1}$ are generated similarly.

5.6. Enhancement on Merging Order

The basic AST-DME algorithm is outlined in Figure 3. In addition to the main ideas described here, this algorithm can be enhanced by two existing techniques.

1. Simultaneous multiple mergings for speed-up. It was pointed out in [5] that multiple subtree pairs can be merged simultaneously instead of merging only one pair each time. The multi-merging scheme can reduce the number of updates on the nearest neighbor graph and therefore reduce the runtime.
2. Delay target based merging order for further wire-length reduction. It was observed in [2] that the wire-length may be affected by the relative delay targets of subtrees in addition to their proximity. A delay target of a node (or a subtree) is a desired delay from clock source to that node (or the root of the subtree). By merging subtrees with large delay targets first, the imbalance on delay targets of subtrees can be reduced. Consequently, the chance of wire snaking is reduced.

Procedure: AST-DME
Input: A set of sink groups $\mathcal{G} = \{G_1, G_2, \dots, G_k\}$ Skew constraints for sinks within each group
Output: A clock routing tree connecting all sinks and satisfying all intra-group skew constraints
<ol style="list-style-type: none"> 1. Initialize a set \mathcal{T} of subtrees with all sinks 2. While $\mathcal{T} > 1$ 3. Find a pair of subtrees $T_a \in \mathcal{T}$ and $T_b \in \mathcal{T}$ with min distance between their roots among all subtrees 4. If both T_a and T_b are from G_i Merge T_a and T_b to T_c at MR_c satisfying skew constraints in G_i 5. Else if T_a and T_b are from different groups Merge T_a and T_b to T_c at MR_c which is the SDR between MR_a and MR_b 6. Else if T_a and T_b share one group Merge based on nearest boundaries of MR_a and MR_b, Merge all sink groups involved with T_a and T_b 7. Else (T_a and T_b share multiple groups) Merge based on intersections of skew bounds induced by merging regions in T_a and T_b, Merge all sink groups involved with T_a and T_b 8. $\mathcal{T} = \mathcal{T} - T_a - T_b + T_c$

Figure 3. The proposed AST-DME algorithm.

Both of the above two techniques can be straightforwardly included in our AST-DME algorithm.

6. Experimental Results

Our algorithm is implemented in C/C++ and the experiments are performed on a Linux system with a Pentium-4 processor of 1.6GHz and 256MB RAM. The benchmark circuits are r1-r5 from [4]. In the experiments, we partition the sinks of each circuit into various number of groups which are intermingled with each other.

We compare our AST-DME algorithm with a simple extension of the greedy-DME algorithm [5]. Originally, greedy-DME is designed for zero skew routing. For the case of associative skew routing, we simply require the skew between different groups to be zero and run the greedy-DME algorithm. The results are listed in Table 1. Both the extended greedy-DME and our AST-DME algorithms can satisfy the skew constraints. The results on wirelength show that our AST-DME consistently outperforms the extended greedy-DME algorithm. Usually, the improvement from AST-DME is more significant when the number of sink groups is increased. The runtime of our algorithm is greater than that of greedy-DME as expected, but still at a reasonable order of magnitude.

7. Conclusion

In this work, we attempt to solve the associative skew clock routing problem especially for the difficult instances where sink groups are intermingled. We find that this problem can be solved well by carefully assembling existing clock routing techniques. Experimental results show that our approach consistently outperforms an extension of a popular conventional method.

References

- [1] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng. Zero skew clock routing with minimum wirelength. *IEEE Transactions on Circuits and Systems - Analog and Digital Signal Processing*, 39(11):799–814, Nov. 1992.
- [2] R. Chaturvedi and J. Hu. An efficient merging scheme for prescribed skew clock routing. *IEEE Transactions on VLSI Systems*, 13(6):750–754, June 2005.
- [3] Y. Chen, A. B. Kahng, G. Qu, and A. Zelikovsky. On the associative-skew clock routing problem. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 168–172, 1999.
- [4] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao. Bounded-skew clock and Steiner routing. *ACM Transactions on Design Automation of Electronic Systems*, 3(3):341–388, July 1998.
- [5] M. Edahiro. A clustering-based optimization algorithm in zero-skew routings. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 612–616, 1993.
- [6] J. P. Fishburn. Clock skew optimization. *IEEE Transactions on Computers*, C-39:945–951, July 1990.
- [7] I. S. Kourtev and E. G. Friedman. Clock skew scheduling for improved reliability via quadratic programming. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 239–243, 1999.
- [8] W.-C. D. Lam, C.-K. Koh, and C.-W. A. Tsao. Power supply noise suppression via clock skew scheduling. In *Proceedings of the IEEE International Symposium on Quality Electronic Design*, pages 355–360, 2002.
- [9] J. L. Neves and E. G. Friedman. Design methodology for synthesizing clock distribution networks exploiting nonzero localized clock skew. *IEEE Transactions on VLSI Systems*, 4(2):286–291, June 1996.
- [10] J. Qian, S. Pallela, and L. T. Pillage. Modeling the effective capacitance for the RC interconnect of CMOS gates. *IEEE Transactions on Computer-Aided Design*, 13(12):1526–1535, Dec. 1994.
- [11] T. Sakurai. Perspectives on power-aware electronics. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 26–29, 2003.
- [12] C.-W. A. Tsao and C.-K. Koh. UST/DME: a clock tree router for general skew constraints. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 400–405, 2000.

Table 1: Comparison between experimental results from an extended greedy-DME algorithm [5] and our algorithm of AST-DME.

Circuit	#groups	Algorithm	Wirelen	Reduction	CPU(s)
r1 267 sinks	1	Greedy-DME	1070421		3
	4	AST-DME	972453	9.2%	11
	6	AST-DME	943334	11.9%	12
	8	AST-DME	930290	13.1%	13
	10	AST-DME	928488	13.3%	13
r2 598 sinks	1	Greedy-DME	2169791		15
	4	AST-DME	1941703	10.5%	34
	6	AST-DME	1939003	10.6%	37
	8	AST-DME	1869138	13.9%	37
	10	AST-DME	1855565	14.5%	37
r3 862 sinks	1	Greedy-DME	2734959		27
	4	AST-DME	2453645	10.3%	41
	6	AST-DME	2370732	13.3%	44
	8	AST-DME	2387000	12.7%	50
	10	AST-DME	2309244	15.6%	48
r4 1903 sinks	1	Greedy-DME	5442046		65
	4	AST-DME	4922289	9.5%	115
	6	AST-DME	4788899	12.0%	119
	8	AST-DME	4825207	11.3%	121
	10	AST-DME	4692357	13.8%	122
r5 3101 sinks	1	Greedy-DME	8033650		146
	4	AST-DME	7249922	9.8%	182
	6	AST-DME	7032847	12.5%	186
	8	AST-DME	6917632	13.9%	192
	10	AST-DME	6897530	14.1%	194

- [13] R.-S. Tsay. Exact zero skew. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 336–339, 1991.
- [14] J. G. Xi and W. W.-M. Dai. Useful-skew clock routing with gate sizing for low power design. *Journal of VLSI Signal Processing*, 16(2/3):163–179, Jun./Jul. 1997.
- [15] H. Zhou. Efficient Steiner tree construction based on spanning graphs. *IEEE Transactions on Computer-Aided Design*, 23(5):704–710, May 2004.