Fast Incremental Link Insertion in Clock Networks for Skew Variability Reduction

Anand Rajaram
ECE Department, University of Texas at Austin
Austin, Tx 78712.
Dallas DSP, Texas Instruments
Dallas, Tx 75243.
anandr@mail.utexas.edu

David Z. Pan
ECE Department,
University of Texas at Austin
Austin, Tx 78712.
dpan@ece.utexas.edu

Abstract

With the advent of sub-100nm VLSI technologies, variation effects greatly increase the unwanted skew in Clock Distribution Networks (CDNs), thereby reducing the performance of the chip. Recent works on link based non-tree CDN [1–4] propose cross-link insertion in a given clock tree to reduce skew variation. However, the current methods suffer from the drawback that they are empirical in nature, requiring the user to experiment with different parameter values. Also, the methods of [1–3] ignore the interaction between the different links while selecting the links for insertion. The method of [4] attempts to overcome this drawback using a statistical link insertion methodology. However, [4] is very slow even for relatively small circuits. In this paper, we propose a fast link insertion methodology which does not require selecting empirical parameters for link insertion. Our method also incrementally considers the effect of previously inserted links before choosing the next link. SPICE based Monte Carlo simulations show that our approach obtains comparable skew reduction to that of the existing approaches while drastically reducing the time taken to obtain a good link-based non-tree CDN.

1 Introduction

One of the most important design challenges faced in the sub 100nm VLSI technologies is the effect of variations such as manufacturing variations, power supply noise and temperature fluctuations on clock network skew. The unwanted skew in the clock network is becoming a significant portion of the total clock time period, thereby limiting the speed and performance of the chip. The link-based CDNs proposed in [1–3] is a promising approach to reduce the skew variability in which cross-links are inserted in a given clock tree at appropriate locations. The cross links increase the correlation between the different sink delays which typically results in reduced skew variability. Figure 1 illustrates the link addition for a general clock tree.

The link selection schemes of [1–3] select the set of links to be inserted in a single step and then add all the links at the same time. Hence, we will refer to all these methods as *one-step* link insertion methods for the rest of this paper. An important drawback of the one-step link insertion methods is that they require the user to experiment with different empirically selected parameter values to get a CDN with reduced skew variability. Also, all the links are added in a single step considering the effectiveness of the individual links *separately*. Thus the effects of the links on each other is ignored. The drawbacks of the one-step link insertion has been discussed in detail in Section 2.1.

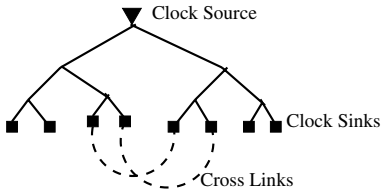

Figure 1. A simple link-based clock network.

Recently, a statistical link insertion methodology has been proposed in [4], in which the interactions between the different links have been considered. However, [4] is very slow even for circuits of small sizes. For example, [4] requires about 10 hours for inserting 20 links in a clock network with 597 sinks. The other drawbacks of the approach of [4] has been explained in detail in Section 2.2.

In this paper, we attempt to solve the problem of incremental link insertion without suffering from the drawbacks of existing methods. SPICE based Monte Carlo simulations show that our method is both effective and very fast when compared to the existing methods.

2 Drawbacks of the Existing Link Insertion Methodologies

We will briefly review existing methods of [1–4] and discuss their limitations. A more detailed explations of the existing methods can be obtained from the corresponding papers. Similar to the works of [1–4], we use Elmore delay for choosing the links to be inserted. However, we verify our methodology using SPICE based Monte Carlo simulations.

2.1 One-step Link Insertion Method and its Drawbacks

2.1.1 Preliminaries

According to [1], if a link is inserted between nodes u and w, the skew between u and w after link insertion is given by:

$$\hat{q}_{u,w} = \frac{R_l}{R_l + r_u - r_w}(q_{u,w} + \frac{C_l}{2}(R_{u,u} - R_{w,w})) \quad (1)$$

where, $q_{u,w}$ is the original skew between nodes u & w, $\hat{q}_{u,w}$ the final skew after the link insertion, C_l the link capacitance, R_l the link resistance, $R_{u,u}$ & $R_{w,w}$ the transfer resistances of nodes u & w. The r_u & r_w are equal to the Elmore delay at u & w when $C_u = 1, C_w = -1$ and the other node capacitance are zero. The C_u and C_w are the total node capacitance at nodes u and w respectively. The above equation can be rewritten as:

$$\hat{q}_{u,w} = (\alpha q_{u,w} + \alpha\beta) \quad (2)$$

where $\alpha = \frac{R_l}{R_l + r_u - r_w}$ and $\beta = \frac{C_l}{2}(R_{u,u} - R_{w,w})$.

From equation 2, it can be observed that the final skew value is the sum of two terms. The first term is the value of the original skew scaled by α. It has been proved in [1] that the value of α is always less than 1. The second term represents the extra skew introduced by the value of link capacitance. If the original value of skew is zero, then it can be seen that adding *only* a resistance still maintains the zero skew. However, adding the link capacitance changes the nominal skew. In [1], the original skew after link insertion is restored by tuning the locations of the internal clock network nodes similar to the method in [8]. This makes sure that the link insertion does not affect the nominal skew of the clock network. It also guarantees that link insertion between sinks u and w always reduces the skew as shown in [1]. It has also been proved in [1] that adding the links at the sink nodes is better for skew reduction than at the top-level nodes. Henceforth, we will assume that the links are inserted only between clock sinks. The major steps in the approach taken by [1] towards building a non-tree CDN are listed below:

1. Given an initial clock tree, select all the node pairs where cross links are to be inserted. The initial clock tree can be obtained from available clock tree routing algorithms in the literature like [6, 7, 9].
2. Add only the link capacitance values at the sink nodes and tune the internal nodes of the clock tree to restore original skew. The tuning is done in a bottom-up fashion similar to the method in [8].
3. Finally, add the link to the selected node pairs. Since the effect of link capacitance has been already removed by bottom-up tuning, only the effect of link resistances will be present.

It has been shown in [1] that this method reduces the skew variability of the clock network considerably when compared to the initial tree with very little increase in wirelength. The key step in the above method is the step of selecting the node pairs for link insertion. In [1, 2] two types algorithms, namely rule-based methods and graph theory based methods, have been proposed for selecting the links. These methods are discussed in the next section.

2.1.2 Rule Based and Graph Theory Based Methods

In the rule-based link insertion methods of [1], all the possible links are characterized by three parameters denoted by α, β and γ. In [2], an additional parameter called δ was introduced. The definitions of α and β are the same as in equation 2. The readers are referred to the [1, 2] for a detailed explanation of the parameter definitions.

The rule-based methods of [1, 2] involve empirically selecting appropriate values of bounds α_{max}, β_{max} , γ_{max} and δ_{max} and adding only the links that have α, β, γ and δ values less than chosen bounds. It may be noted here that a link with least values of α, β, γ and δ is usually better [1]. But selecting too small values of the bounds will result in very few links getting added, resulting in less skew variation reduction.

The graph theory based link insertion methods of [1, 2] involve appropriately dividing the clock tree into recursive bi-partite graphs and choosing the number of links to be inserted based on edge weights of the bi-partite graph. The edge weights are proportional to the wirelength of the links considered. The readers are referred to [1, 2] for a detailed discussion on graph theory methods.

2.1.3 Drawbacks of the One-step Link Insertion

One important drawback of the one-step link insertion algorithm is that the effect of the links on each other is not considered. For example, in Figure 2, let us assume that $link\ a$ is already selected for insertion and that before the insertion of $link\ a$, the $link\ c$ had a lower α value when compared to $link\ d$. As a result, the one-step algorithms will choose $link\ c$ over $link\ d$. However, after inserting $link\ a$, the delays of all the nodes become correlated and the delay of every sink in the clock network is at least loosely

correlated with the delays of every other sink. This is because, after inserting the first link $link\ a$, the values of r_u and $R_{u,u}$ used in equation 2 will change for every sink, resulting in changed the values of the α and β for all possible links. Hence, the new values of α for $link\ d$ might be lower than that of $link\ c$ as a result of which it might be better to choose $link\ d$. In other words, even though $link\ c$ is better than $link\ d$ for the initial clock tree, $link\ d$ might become better after the insertion of the first link. As a result, any methodology that considers the effect of the previously selected links before selecting the next link might perform better in skew variability reduction.

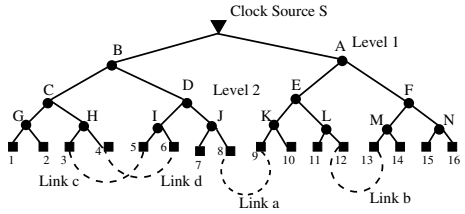

Figure 2. A simple nontree where incremental approach might help.

Another drawback of the rule-based methods is that they require the user to empirically select the values of different parameters like α_{max}, β_{max} , γ_{max} and δ_{max}. Similarly, in the case of the graph theory based methods, the only physical characteristic of the link considered is the length of the link. However, for similar reasons as outlined in the previous paragraph, the link with minimal size might not be the best link to be added after an initial set of links have been inserted. Though the graph theory based methods does not require explicit parameters like the rule-based methods, the user still needs to select the number of links to be inserted at each level of the clock network. Thus, all the one-step methods require the user to experiment with different non-trees before selecting a good non-tree. As a result, the amount of time that the user might have to spend to obtain the appropriate values depends on the specific test case. An incremental link insertion method that does not involve empirical selection of parameters will address these drawbacks.

2.2 Drawbacks of Statistical Link Insertion of [4]

The statistical link insertion method of [4] adds the links incrementally by updating the statistical values of the sink delays and using the information of the latest non-tree to select the next link to be inserted. This method has the advantage that every link insertion considers the effects of all the previously inserted links, unlike the methods of [1–3]. However, the method of [4] suffers from the serious disadvantage that the runtime is very high even for clock networks of relatively small size. Also, the increase in runtime as a function of the clock network size is exponential. For example, clock networks with 74, 179 and 597 sinks take roughly 3, 50 and 655 minutes respectively [4]. As a result, the method of [4] cannot be used for bigger clock networks. Another point to be noted here is that the skew variation reduction as a result of link insertion is independent of the source of skew. This can be easily verified from Equation 2 in which the skew $q_{u,w}$ gets scaled down by the value of α irrespective of the source of $q_{u,w}$. Thus, considering all the independent variables like wirewidths, load capacitances etc. (as is done in [4]) for link insertion does not bring any definite advantage but it will increase the amount of computation drastically. Thus, a fast incremental link insertion method that considers the effects of the previously selected links will address these drawbacks.

3 Fast Incremental Link Insertion

In this section, we propose the incremental link insertion in which we have attempted to address some of the disadvantages of existing methods. Our objectives are:

1. The *effect of previously inserted links* on variability reduction must be considered before selecting the next link. This might help in selecting better places for link insertion.
2. *Automatic link selection* without requiring the users to empirically select the parameters is required. This will allow for a complete automation of link insertion and save the time spent in selecting the appropriate parameters.
3. Fast and Efficient incremental link insertion method will be required. This will allow us to insert links even in bigger clock networks.

Please note that none of the currently available link insertion methods can satisfy the above requirements.

3.1 Automatic Link Selection

In this subsection, we will discuss our method of fully automated link insertion. Consider the Figure 2. The dashed lines represent the links and the solid lines represent the given clock tree. In the original tree, the nominal Elmore delay (w.r.t. the output node of buffer S) of sinks 1 and 16 are practically independent of each other. However, if the $link\ a$ is added between the sinks 8 and 9, even the nominal delays of sinks 1 and 16 become somewhat correlated due to the fact that both the sinks now have more than one source to sink paths and both sinks share the $S \rightarrow A$ and $S \rightarrow B$ paths. It is because of this correlation that the skews of a link based non-tree due to random variation gets reduced.

Among the links shown in Figure 2, $link\ a$ is usually the best because it links two sinks that are hierarchically

farthest apart and physically closest. After the insertion of $link\ a$, the delays of all sinks are correlated with the delays of all the other sinks. The best position to insert a link will be in such a place where the correlation between the sink delays further increases. To the best of our knowledge, the only way to reliably identify the correlation between the different sink delays is to perform Monte Carlo Analysis. This is because of the absence of any closed form solution to express the correlation between the delays of the sinks of a general (non-tree) RC structure. However, using Monte Carlo analysis to select each link might be time consuming.

We propose to overcome this problem as follows. From the definition of α in equation 2, we can observe that a link with the least value of α is usually among the best links to insert because α measures the scale by which the original skew gets reduced. Because of this, α value can be used as a measure of selecting the links. However, obtaining the α value for all possible links for a given non-tree is non-trivial and will require the computation of Elmore Delay for each sink pair (u, w) with $C_u = +1$ and $C_w = -1$. Thus, for n sinks, $O(n^2)$ evaulation of Elmore delays is necessary. Though the methods in [1, 2] make use of the α parameter, they obtain the α values only for the initial clock tree and they *never update* the values of r_u and r_w used in equation 1. The initial values of r_i for any node i will be equal to the sum of all the resistance in the source$\rightarrow i$ path. After inserting of the first set of links, because of the existence of multiple source to sink paths, there is no closed form expression to obtain the correct values of r_i. Thus, [1, 2] uses only the initial α values of the links which might be grossly different from the values after inserting even the first link. If we can find a method to efficiently update the values of α for all possible links after each link insertion and select the next link based on the latest clock network structure, it will solve our objective 1. We describe such a method in Section 3.2 in which we make use of an a term called *equivalent* r_i, denoted by E_r_i which we use to evaluate the value of α defined in equation 2.

It may be noted here that we update only the nominal values of circuit parameters, unlike [4] in which the statistical values like mean, variance are updated. By using only the nominal circuit delay values, we make use of the fact that skew reduction due to link addition is independent of the source of skew. Hence, we avoid time consuming steps such as the statistical method of [4] or Monte Carlo simulations to select links. Our objective 2 can be met by adding only the link with the least value of α each time and the stopping criterion can be based on the maximum wire length increase that the user can accept.

3.2 The Algorithm

Figure 3 shows our top-level algorithm, which is an incremental equivalent to the one-step algorithms of [1, 2]. The three main subroutines of the top level algorithm are $Pick\_Best\_Link$, $Node\_Tune$ and $Update\_Equivalent\_r_i\_Values$. The procedure $Pick\_Best\_Link$ picks the best link for a given non-tree (or the initial tree) based on the *latest* values of α. The $Update\_Equivalent\_r_i\_Values$ procedure *efficiently* obtain the correct values of α parameter of the latest non-tree. A point that may be noted here is that the method of [4] does not eliminate the adverse effect of the link capacitance after selecting a particular link. In this work, we use the $Node\_Tune$ procedure to efficiently eliminate the negative effect of link capacitance. As a result, the scaling effect of each link on unwanted skew can be expected to be much higher when compared to the method of [4]. This fact can be verified from equation 2 in which the link capacitance might increase the final skew.

Procedure: $Insert\_Links(CDN_s)$
Inputs: Clock Network CDN_s rooted at source s, MLC = Max_Link_Cost.
Initialization: Set L = {}, Link_Cost = 0.
Output: Link node pair set L.
1. $link = Pick\_Best\_Link(CDN_s)$.
2. If $Link\_Cost + len(link) > MLC\ Go\ To\ 8$.
3. $L \leftarrow L + link; Link\_Cost+ = len(link)$.
4. Add link capacitance to selected node-pair n1,n2.
5. $Node\_Tune(CDN_s, n1, n2)$ to restore skew.
6. Add link resistance between nodes $n1, n2$.
7. $Update\_Equivalent\_r_i\_Values(CDN_s, link)$. $Go\ To\ step\ 1$.
8. Return L

Figure 3. The top-level incremental link insertion algorithm.

The $Pick\_Best\_Link$ procedure used in step 1 of Figure 3 is described in Figure 4. The best link picked by the $Pick\_Best\_Link$ procedure is added only when adding it does not increase the wirelength consumption of the clock network beyond a user-set wirelength bound. This is shown in the steps 2 and 3 of the Figure 3. In the step 4 of Figure 3, the link capacitances are added to the two sink nodes after which the $Node\_Tune$ step tunes the locations of the internal nodes to restore the original skew. The $Node\_Tune$ step is very similar to the method in [8] where the internal node locations are tune to obtain a zero skew clock tree.

After the internal nodes have been tuned in the step 5 of the top-level algorithm, steps 6 and 7 add the link resistance and then update the values of E_r_i for each node i using the $Update\_Equivqlent\_r_i\_Values$ subroutine. The different steps of the $Update\_Equivqlent\_r_i\_Values$ subroutine are described below.

1. Remove all link resistances from the CDN_S, includ-

Procedure: $Pick\_Best\_Link(CDN_s)$.
Input: Clock Network CDN_s rooted at source s. **Output:** The best link based on updated α value of the non-tree(or the initial tree)
1. $l \leftarrow$ left child node of s. 2. $r \leftarrow$ right child node of s. 3. For all pairs $(i, j), i \in$ sinks(l) and $j \in$sinks(r). $link$ = Link node pair (i, j) with min value of α. 4. Return $link$

Figure 4. The method of picking the best link based in α value of the non-tree.

ing the previously added links. This will make the structure a tree with only the link capacitances.

2. Set the values of all sink node capacitances to be +1 and all the internal node capacitances to be zero.
3. Evaluate the Elmore delays for all the sink nodes in the clock tree.
4. Using this initial Elmore delay values and the iterative procedure of [5], evaulate the final Elmore delays by iteratively adding all the link resistances. The final value of the Elmore delays of the sink nodes i gives the value of the *equivalent* r_i, denoted by E_r_i for each sink i.

The key idea in the above procedure is in the step 2 in which we set the values of all the sink capacitance to be +1 and all other internal node capacitance to be 0. Please note that the values of resistances are still maintained at their original values. The reasoning behind this procedure is explained below. We know that, for a tree structure, the value of r_i (used in equation 1) for any node i equals the total value of resistance of the $source \rightarrow i$ path. However, such a simple evaluation is not possible for a non-tree and we would like to get a similar parameter that captures the effect of equivalent resistance between the source and the sink node i for any given non-tree. One possible parameter that satisfies this requirement is the value of Elmore delay to a sink node i with all the sink capacitances values set to 1 and other capacitances set to zero. The Elmore delay so evaluated will be a weighted sum of terms involving the resistances of *all the* $source \rightarrow i$ *paths*. This value of Elmore delay will enable us to capture the effect of *equivalent* resistance between the source and the node i in an efficient manner.

The main advantage of the above procedure is that, for a given non-tree, a single evaluation of Elmore delays using the method of [5] will enable us to obtain the values equivalent to that of E_r_i value of every sink. Once we have the E_r_i values for all the sink nodes i, we can evaluate the α value of each possible link quickly using the E_r_i values instead of r_i values. This advantage is made use of in the $Pick\_Best\_Link$ procedure. Thus, we can efficiently obtain the values of α for all possible links in a given clock network.

Given an initial clock tree and an upper bound on the total wire-length that the user wants to add, our top-level procedure incrementally chooses the best link based on the α value, tunes the locations of the internal nodes to eliminate the adverse effects of the link capacitance, obtains the values of E_r_i for all nodes i and uses them in the next iteration to choose the next best link. Thus we see that our incremental approach has addressed all the three objectives that we discussed in Section 3. In the next section, we compare the effectiveness of our approach with the existing methods using SPICE based Monte Carlo simulations.

4 Experimental Results

In this section, we compare the skew variability reduction of our incremental approach with the existing approaches in [1, 2] and [4]. From the experimental results of [1–4], we can see that the methods of [2] reports the maximum reduction in skew variability with minimal increase in wirelength. So, we compare the efficacy of our method with the methods of [2]. To facilitate this comparison, our experimental setup is identical to that of [2], i.e., r1-r5 clock tree benchmarks obtained from GSRC Bookshelf [10]. The variation factors considered in our experiments are also identical to [2] namely, the clock driver resistance, wire width and the load capacitance of all sinks. The driver resistance, wire width and the sink capacitance have $\pm 15\%$ variation following a normal distribution. We use HSPICE based Monte Carlo simulations for validating our results. For all the clock networks, a Monte Carlo simulation of 1000 trials is performed using HSPICE. We obtain the worst case skew (WCS) value and the standard deviation (SD) values of skew variations using HSPICE. The size of benchmark circuits, skew variations and wire length of clock trees are given in Table 1.

Table 1. The HSPICE Worst Case Skew in ps (WCS), standard deviation (SD) and total wire length of trees.

Testcase	# sinks	WCS	SD	wirelen	CPU(s)
r1	267	131	31	1320665	1
r2	598	406	79	2602908	3
r3	862	457	109	3388951	4
r4	1903	1390	380	6828510	12
r5	3101	3270	798	10242660	18

Table 2 compares the skew variability reduction of our incremental link insertion and that of the methods of [2]. All the skew values shown in Table 2 are w.r.t. the variation results of the trees in Table 1. Since the method of [2] involves empirical selection of parameters, we have written

a Perl script that will search the solution space of the parameters α, β, γ and δ and select a good link based non-tree using Elmore delay based Monte Carlo simulations. At each step, the script will select a particular set of values for the parameters, add the links and evaluate the skew variability using 100 trials of Monte Carlo simulation. The non-tree with the best skew reduction is selected and simulated using HSPICE using 1000 trials of Monte Carlo simulations. The results so obtained for the rule-delta (RD) and MST methods of [2] is given in the table 2, denoted by Auto-RD and Auto-MST respectively. All the values of skew and wirelength are w.r.t. the tree values shown in table 1. The #Trials column shows the number of non-trees evauluated to select the final non-tree.

Table 2. The Skew variability for non-trees w.r.t. the tree values of Table 1.

Testcase	WCS	SD	wirelen	#Trials	CPU(s)
Auto-RD-r1	0.12	0.11	1.11	120	180
Auto-MST-r1	0.12	0.11	1.08	16	24
Incremental-r1	0.18	0.17	1.10	1	0.5
Auto-RD-r2	0.14	0.12	1.04	180	1260
Auto-MST-r2	0.14	0.12	1.06	16	112
Incremental-r2	0.18	0.2	1.06	1	3.0
Auto-RD-r3	0.12	0.13	1.09	150	3750
Auto-MST-r3	0.13	0.13	1.05	18	450
Incremental-r3	0.12	0.13	1.05	1	6.0
Auto-RD-r4	0.11	0.12	1.05	170	10540
Auto-MST-r4	0.10	0.12	1.02	24	1488
Incremental-r4	0.13	0.14	1.02	1	12.0
Auto-RD-r5	0.09	0.07	1.03	360	86400
Auto-MST-r5	0.07	0.07	1.03	36	8640
Incremental-r5	0.09	0.07	1.02	1	70

The important observations from Table 2 are as follows:

- The incremental link insertion achieves comparable reduction in skew variability to that of the existing algorithms with comparable increase in wirelength consumption.
- The incremental link insertion is very fast even for the biggest benchmark with 3100 sinks. Thus, our method is more efficient that the statistical link insertion of [4]. It is also much faster when compared to the automated version of the algorithms of [2].
- Since the incremental link insertion is a one-shot approach, a good non-tree is obtained in a single trial, thus avoiding the time comsuming manual parameter selection and statistical delay updation methods.

5 Conclusion

A fast and efficient incremental link insertion methodology for clock skew variability reduction has been proposed. The different drawbacks of the existing approaches have been discussed. We attempt to address these drawbacks by proposing the incremental link insertion algorithm. SPICE based Monte Carlo simulations show that the incremental link insertion is both fast and effective when compare with the existing methods. Since incremental link insertion is fully automatic, the time consuming steps like manual parameter selection, Monte Carlo simulations and statistical delay updation are avoided.

6 Acknowledgements

This work is partially supported by SRC and IBM Faculty Award.

References

[1] A. Rajaram, J. Hu, and R. Mahapatra, "Reducing clock skew variability via cross links," in *Proc. of the DAC*, San Diego, CA, pages 18–23, June 2004.

[2] A. Rajaram, D. Z. Pan, and J. Hu, "Improved Algorithms for Link-Based Non-Tree Clock Networks for Skew Variability Reduction," in *Proc. of the ISPD*, San Francisco, CA, pages 55–62, April 2005.

[3] G. Venkataraman, N. Jayakumar, J. Hu, P. Li, S. Khatri, A. Rajaram, P. McGuinness, and C. Alpert, "Practical Techniques for Minimizing Skew and Its Variation in Buffered Clock Networks," in *Proc. of the ICCAD*, San Jose, CA, pages 592-596, November 2005.

[4] W. D. Lam, Y. Chen, J. Jain, C.-Kok. Koh, and V. Balakrishnan, "Statistical Based Link Insertion for Robust Clock Network Design," *Proc. of the ICCAD*, San Jose, CA, pages 587-590, November 2005.

[5] P. K. Chan and K. Karplus, "Computing signal delay in general RC networks by tree/link partitioning," in *IEEE Transactions on CAD*, vol.9, no.8, pages 898–902, August 1990.

[6] C.-W. A. Tsao and C.-K. Koh, "UST/DME: a clock tree router for general skew constraints," in *Proc. of the ICCAD*, San Jose, CA, pages 400–405, November 2000.

[7] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng, "Zero skew clock routing with minimum wirelength," in *IEEE Transactions on CS-ADSP*, vol.39, no.11, pages 799–814, November 1992.

[8] R.-S. Tsay, "Exact zero skew," in *Proc. of the ICCAD*, Santa Clara, CA, pages 336–339, November 1991.

[9] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao, "Bounded-skew clock and Steiner routing," in *ACM Transactions on DAES*, 3(3): pages 341–388, July 1998.

[10] http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/BST/