ε-Optimal Minimum-Delay/Area Zero-Skew Clock Tree Wire-Sizing in Pseudo-Polynomial Time[*]

Jeng-Liang Tsai, Tsung-Hao Chen
Electrical and Computer Engineering
University of Wisconsin-Madison
1415 Engineering Drive
Madison, WI 53706
{jltsai, tchen}@cae.wisc.edu

Charlie Chung-Ping Chen[†]
Graduate Institute of Electronics Engineering &
Department of Electrical Engineering
National Taiwan University
Taipei 106, Taiwan
cchen@cc.ee.ntu.edu.tw

ABSTRACT

In 21st-Century VLSI design, clocking plays crucial roles for both performance and timing convergence. Due to their non-convex nature, optimal minimum-delay/area zero-skew wire-sizing problems have long been considered intractable. None of the existing approaches can guarantee optimality for general clock trees to the authors' best knowledge. In this paper, we present an ϵ-optimal zero-skew wire-sizing algorithm, *ClockTune*, which guarantees zero-skew with delay and area within ϵ distance to the optimal solutions in pseudo-polynomial time. Extensive experimental results show that our algorithm executes very efficiently in both runtime and memory usage. For example, *ClockTune* takes less than two minutes and 35MB memory to size an industrial clock tree with 3101 sink nodes within 2% to the optimal solution on a 533MHz Pentium III PC. Our algorithm can also be used to achieve useful clock skew to facilitate timing convergence and to incrementally adjust clock tree for design convergence and explore delay/power tradeoffs during design cycles. *ClockTune* is available on the web [13].

Categories and Subject Descriptors

B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids

General Terms

Algorithms

Keywords

Zero-skew, clock tree, wire-sizing, incremental refinement, ϵ-optimal, pseudo-polynomial

[*]This work was partially supported by the National Science Foundation under grants CCR-0093309 and CCR-0204468.

[†]This work was done when the author was at department of Electrical and Computer Engineering, University of Wisconsin-Madison.

1. INTRODUCTION

As the feature size keeps shrinking and clock frequency increasing, clock design has come to play a crucial role in determining the chip performance and facilitating timing and design convergence. First, clock skew directly affects chip performance in close to a one-to-one ratio since it has to be counted as cycle time penalty. Second, incremental clock tree adjustment enables fast design convergence by avoiding the potentially divergent design iterations [1]. Since designs are subjected to change on a daily basis, the clock trees need to be incrementally adjusted accordingly with minimum changes to ensure acceptable clock skew. Third, since interconnect delay dominates over gate delay, timing plans often cannot be met due to some physical effects. Recently, useful-skew [2] concepts have also been widely proposed to speed up timing convergence in order to compensate for the timing uncertainties resulting from physical layout. From the above analysis, it is crucial to develop clock tuning algorithms that can balance clock skew with minimum adjustments. One of the most effective ways to adjust clock skew is through wire-sizing since it involves minimum routing modifications.

In [3], discrete wire-sizing for general routing-trees is assumed and a bottom-up dynamic programming approach is used to propagate optimal wire-sizing combinations toward the root node. The designer can then determine the wire-sizing by making a tradeoff between delay and power consumption. In [4] [5], the wire-sizing problem is formulated as optimization problems, in which the maximum delay of each sink node is constrained, and the delay can be minimized efficiently. These methods perform wire-sizing without modifying the clock routing, but do not guarantee zero-skew.

Recent works [6] [7] integrate wire-sizing into the Deferred-Merging Embedding (DME) algorithm [8], which allow a zero-skew clock tree to benefit from wire-sizing. However, the zero-skew property is maintained by moving the merging points, and the clock routing might be changed to accommodate the skew caused by design changes, which might affect the detail routing and there is no guarantee of optimality for these algorithms. In conclusion, there is no optimal wire-sizing algorithm which can guarantee the use of minimum power or area to simultaneously achieve minimum delay and zero-skew to the best of authors' knowledge.

In this paper, we propose a novel wire-sizing algorithm, *ClockTune*, which guarantees zero-skew with a given delay and power consumption solution within ϵ distance to the optimal solution in psuedo-polynominal time. *ClockTune* first calculates the feasible delay and capacitance load information for each node in a bottom-up fashion. After the desired delay and capacitance load is chosen from the feasible region, a wire-sizing is determined in a top-down fashion.

To achieve the ϵ-optimality we use sampling techniques to capture the whole design space and prune out infeasible solutions. Our algorithm takes pseudo-polynomial time, and experimental results show that our algorithm can efficiently re-size large clock trees. Although we focus on achieving zero-skew, *ClockTune* can also be used to achieve useful-skew to tackle timing problems.

The rest of this paper is organized as follows. In Section 2, we formulate the problem and introduce the framework of our algorithm. In Section 3, we derive the formulae and give more details of the algorithm. Section 4 contains the complexity and optimality analyses. Experimental results are shown in Section 6, and Section 7 is our conclusion.

2. PRELIMINARY

Assuming the Elmore delay under π-model [9], the wire-sizing problem is defined as below.

Problem Definition 1. *Minimum-Delay / Area Zero-Skew Wire-Sizing (min-ZSWS) Problem*
Given a clock tree T, find a set of feasible wire widths that achieves the target delay/area such that the zero-skew constraint is satisfied and the area/delay is also minimized.

Table 1 lists the notations used throughout this paper. In Table 1, T_v is a binary tree. However, if the wire length is allowed to be zero, then any tree structure can be represented as a binary tree.

T_v	A clock tree with given routing rooted at node v
v_l	The left child of node v
v_r	The right child of node v
e_v	The edge between node v and its parent node
l_v	The length of edge e_v
$w(e_v)$	The wire width of edge e_v
$r(e_v)$	The resistance of wire e_v, $r(e_v) = \frac{l_v r_0}{w(e_v)}$
$c(e_v)$	The capacitance of wire e_v, $c(e_v) = l_v w(e_v) c_0$
$w_m(e_v)$	Minimum feasible wire width for edge e_v
$w_M(e_v)$	Maximum feasible wire width for edge e_v
d_v	Elmore delay from node v to any leaf node in T_v
c_v	Total down-stream capacitance seen at node v
r_0	Wire resistance per μm^2
c_0	Wire capacitance per μm^2
w_m	Minimum wire width defined by user
w_M	Maximum wire width defined by user

Table 1: Notations for the min-ZSWS problem

2.1 The concept of DC Region

For most general routing-tree wire-sizing problems, only the maximum delay is constrained. Due to the convex nature of these problems, bottom-up approaches that keep the lower-left portion of the delay-area tradeoff curve can solve these problems optimally. On the contrary, the min-ZSWS problem is non-convex and the whole design space has to be kept to solve the problem optimally.

A set of wire widths which makes the clock tree T_v zero-skew is an *embedding*. Each embedding determines a pair of delay value and total capacitance load value of T_v. The former relates to the clock skew due to process variation, and the latter is proportional to the dynamic power consumption, $P_d = fCV^2$.

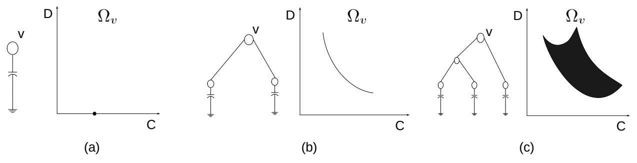


Figure 1: DC regions at node v, where v is a (a) leaf node (b) level-1 node (c) level-2 node

The entire design space can be captured by mapping all embeddings on the *D-C plane*, the X-Y plane with delay value on Y-axis and capacitance load value on X-axis.

Definition 1. *DC Region*
The DC region of node v, $\Omega_v = \{(d_v, c_v)\}$, is a set of points on the D-C plane which represents all feasible embeddings of T_v.

The DC region represents the solution space mapped on the D-C plane. Figure 1 gives examples for different types of DC regions. For level-2 and above nodes, the optimal solutions lie on the lower-left edge of the DC region. However, in dealing with the min-ZSWS problem, pruning out sub-optimal solutions during bottom-up propagation might lead to failure in finding a solution at the root node. Moreover, incremental refinement is not possible if only optimal solution sets are preserved because the optimal solution sets are likely to become infeasible after design changes. Thus new algorithms that guarantee optimality and are able to achieve incremental refinement must be able to capture the whole DC region.

2.2 Algorithm overview

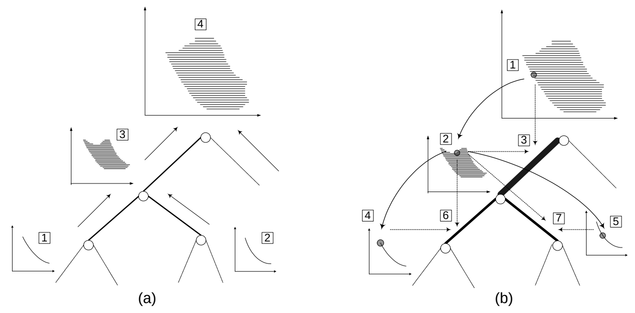


Figure 2: Illustration of *ClockTune* (a) the bottom-up phase (b) the top-down phase

We propose a dynamic programming algorithm, *ClockTune*, to solve the min-ZSWS problem. *ClockTune* is composed of two phases. In the first phase, a bottom-up approach is used to obtain the DC regions of all nodes. In the second phase, a top-down approach determines the widths of all wire segments. The dynamic programming algorithm *ClockTune()* is given as a pseudocode in Algorithm 1, and Figure 2 illustrates its procedures.

Algorithm 1 *ClockTune(T_v)*

```
Input: a clock tree $T_v$ with given routing rooted at node $v$
Output: an embedding of $T_v$
    $\Omega_v \leftarrow$ ClockTune_DC($T_v$)
    /* bottom-up construct DC regions, detailed in 3.1 */
    Choose $(d_t, c_t)$ from $\Omega_v$
    /* choose target delay and capacitance load for $T_v$ */
    Call ClockTune_Embed($d_t, c_t, T_v$)
    /* top-down embedding selection, detailed in 3.2 */
```

3. ε-OPTIMAL ZERO-SKEW WIRE-SIZING ALGORITHM

In this section, the details of *ClockTune* are described. In 3.1, the procedures to obtain the DC regions of different types of nodes are examined. It is then followed by the top-down wire-sizing procedure in 3.2.

3.1 Bottom-up Phase

In this subsection, we first introduce the definition of *branch DC region* and the associated $\curlywedge$ operator to facilitate our discussion. The calculation of DC regions for each type of node is then followed.

Definition 2. *Branch DC Region*
The branch DC region of node v, $\Omega_v^+ = \{(d_v^+, c_v^+)\}$, is a set of points on the D-C plane which represents all feasible embeddings of $T_v^+ = \{e_v \cup T_v\}$ such that d_v^+ and c_v^+ are the Elmore delay and capacitance load seen at the root node of T_v^+. Ω_v^+ can be obtained from Ω_v by the transformation $\mathbb{M}_v : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ described as follows.

$$\begin{aligned} d_v^+ &= d_v + \frac{l_v r_0}{w(e_v)} \left(\frac{l_v w(e_v) c_0}{2} + c_v \right) \\ c_v^+ &= c_v + l_v w(e_v) c_0, \\ & \quad \textit{where } (d_v, c_v) \subset \Omega_v, w_m \le w(e_v) \le w_M \end{aligned}$$

Definition 3. $\curlywedge$ *Operator*
The DC region of v can be generated from the branch DC regions of v_l and v_r by $\curlywedge$ operator. The operation is defined as follows:

$$\begin{aligned} &\Omega_v = \Omega_{v_l}^+ \curlywedge \Omega_{v_r}^+, where \\ &(d_v, c_v) \subset \Omega_v \iff \exists (d_{v_l}^+, c_{v_l}^+) \subset \Omega_{v_l}^+, (d_{v_r}^+, c_{v_r}^+) \subset \Omega_{v_r}^+ \\ &\qquad s.t. \;\; d_v = d_{v_l}^+ = d_{v_r}^+, c_v = c_{v_l}^+ + c_{v_r}^+. \end{aligned}$$

The $\curlywedge$ operation combines all feasible embeddings with same delays on both subtrees. Therefore, the DC region of the root node can be obtained by recursion. To use more accurate delay models such as AWE [10] method, one can use more complex transformations to obtain Ω_v^+ from Ω_v, or use a fudge factor approach to approximate the exact delay and capacitance load [11]. *ClockTune_DC()* is given in Algorithm 2 and the details on how to obtain the DC regions for each type of nodes are given in the following subsections.

3.1.1 Leaf Nodes with Zero-Skew and Useful-Skew Constraints

For a leaf node v, c_v is the load capacitance and hence a constant. If zero-skew is desired, we can set d_v to 0 for all leaf nodes. $\Omega_v = \{(d_v, c_v)\}$ is a single point on the D-C plane. In designing critical components, time borrowing

Algorithm 2 *ClockTune_DC(T_v)*

```
Input: a clock tree $T_v$ with given routing rooted at node $v$
Output: DC regions of all nodes in $T_v$
if $v$ is a leaf node then
  $\Omega_v \leftarrow (d_v, c_v)$
else {$v$ is an internal node}
  call ClockTune_DC($T_{v_l}$)
  call ClockTune_DC($T_{v_r}$)
  switch{$v$}
    case level-1 node
      obtain $\Omega_v$ from $\Omega_{v_l}$ and $\Omega_{v_r}$ by 3.1.2
    case level-2 node
      obtain $\Omega_{v_l}^+$ from $\Omega_{v_l}$ and $\Omega_{v_r}^+$ from $\Omega_{v_r}$ by 3.1.3
      $\Omega_v \leftarrow \Omega_{v_l}^+ \curlywedge \Omega_{v_r}^+$
    case level-3 or above node
      obtain $\Omega_{v_l}^+$ from $\Omega_{v_l}$ and $\Omega_{v_r}^+$ from $\Omega_{v_r}$ by 3.1.4
      $\Omega_v \leftarrow \Omega_{v_l}^+ \curlywedge \Omega_{v_r}^+$
  end switch
end if
```

between consecutive logic blocks is achieved by useful-skew. Useful-skews are often considered in the early design stage, however, they might also be used to fix the timing problems for free. For example, if there is a path timing failure but there are still timing budgets left in the prior or next stage, useful-skews can be introduced to fix the timing problem. Although our focus is on the min-ZSWS problem, *ClockTune* can accept arbitrary skew values by simply assigning a different d_v to each leaf node and find an embedding if it exists. This is because *ClockTune* operates on DC regions, and whether the leaf nodes require zero-skews or useful-skews does not make any difference. *ClockTune* can also be extended to accept bounded-skew constraints, where Ω_v becomes a vertical segment. If intentional skew is desired, [2] [12] can be used for initial routing.

3.1.2 Level-1 Nodes

A level-1 node v has two leaf children, v_l and v_r. We know that

$$d_{v_l}^+ = d_{v_l} + \frac{l_{v_l}^2 r_0 c_0}{2} + \frac{l_{v_l} r_0 c_{v_l}}{w(e_{v_l})} \quad (1)$$

$$d_{v_r}^+ = d_{v_r} + \frac{l_{v_r}^2 r_0 c_0}{2} + \frac{l_{v_r} r_0 c_{v_r}}{w(e_{v_r})}. \quad (2)$$

Observing zero-skew and wire width constraints, we have the following equations.

$$d_v = d_{v_l}^+ = d_{v_r}^+ \quad (3)$$

$$w_m \le w(e_{v_l}) \le w_M \quad (4)$$

$$w_m \le w(e_{v_r}) \le w_M \quad (5)$$

From (1)(2)(3), we can derive the relation between $w(e_{v_l})$ and $w(e_{v_r})$.

$$w(e_{v_r}) = \frac{l_{v_r} r_0 c_{v_r}}{(d_{v_l} - d_{v_r}) + \frac{r_0 c_0}{2}(l_{v_l}^2 - l_{v_r}^2) + \frac{l_{v_l} r_0 c_{v_l}}{w(e_{v_l})}} \quad (6)$$

Thus, $\Omega_v = \{(d_v, c_v)\}$ can be written with a single variable

$w(e_{v_l})$ as

$$d_v(w(e_{v_l})) = d_{v_l} + \frac{l_{v_l}^2 r_0 c_0}{2} + \frac{l_{v_l} r_0 c_{v_l}}{w(e_{v_l})}, \quad (7)$$

$$c_v(w(e_{v_l})) = c_{v_l} + c_{v_r} + l_{v_l} w(e_{v_l}) c_0 + \frac{l_{v_r}^2 c_{v_r} r_0 c_0}{(d_{v_l} - d_{v_r}) + \frac{r_0 c_0}{2}(l_{v_l}^2 - l_{v_r}^2) + \frac{l_{v_l} r_0 c_{v_l}}{w(e_{v_l})}}. \quad (8)$$

From (4) (5) (6), the range of $w(e_{v_l})$ and $w(e_{v_r})$ are

$$max\left(\frac{l_{v_r} r_0 c_{v_r}}{(d_{v_l} - d_{v_r}) + \frac{r_0 c_0}{2}(l_{v_l}^2 - l_{v_r}^2) + \frac{l_{v_l} r_0 c_{v_l}}{w_m}}, w_m\right) = w_m(e_{v_r}) \le w(e_{v_r}) \le w_M(e_{v_r}) = min\left(\frac{l_{v_r} r_0 c_{v_r}}{(d_{v_l} - d_{v_r}) + \frac{r_0 c_0}{2}(l_{v_l}^2 - l_{v_r}^2) + \frac{l_{v_l} r_0 c_{v_l}}{w_M}}, w_M\right); \quad (9)$$

$$max\left(\frac{l_{v_l} r_0 c_{v_l}}{(d_{v_r} - d_{v_l}) + \frac{r_0 c_0}{2}(l_{v_r}^2 - l_{v_l}^2) + \frac{l_{v_r} r_0 c_{v_r}}{w_m}}, w_m\right) = w_m(e_{v_l}) \le w(e_{v_l}) \le w_M(e_{v_l}) = min\left(\frac{l_{v_l} r_0 c_{v_l}}{(d_{v_r} - d_{v_l}) + \frac{r_0 c_0}{2}(l_{v_r}^2 - l_{v_l}^2) + \frac{l_{v_r} r_0 c_{v_r}}{w_M}}, w_M\right). \quad (10)$$

From (7) (8), the first derivative is always negative for d_v and always positive for c_v, thus

$$d_v(w_M(e_{v_l})) \le d_v \le d_v(w_m(e_{v_l})), \quad (11)$$
$$c_v(w_m(e_{v_l})) \le c_v \le c_v(w_M(e_{v_l})). \quad (12)$$

Equations (7) (8) with boundary conditions (11) (12) completely capture Ω_v.

3.1.3 Level-2 Nodes

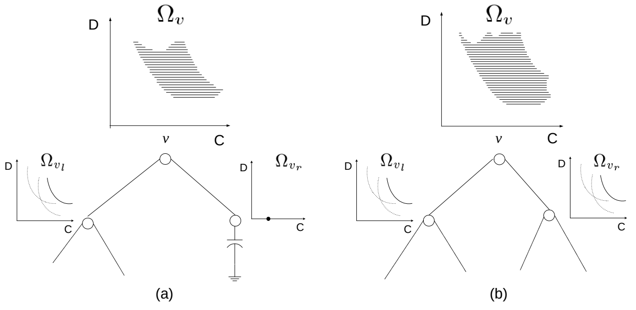


Figure 3: DC regions of Level-2 nodes

As shown in Figure 3, the DC regions of level-2 nodes become more complex and are difficult to solve analytically. To overcome this problem, we use line sampling technique to calculate and store the DC region for level-2 and above nodes. The first step is to find out the upper and lower delay bounds of Ω_v.

Calculate Delay Range

A Level-2 node v has at least one level-1 child. Assuming v_l is a level-1 node and $w_a = w(e_{v_{ll}})$, $\Omega_{v_l}^+$ can be derived from (7)(8) as

$$d_{v_l}^+(w(e_{v_l}), w_a) = d_{v_l}(w_a) + \frac{l_{v_l}^2 r_0 c_0}{2} + \frac{l_{v_l} r_0 c_{v_l}(w_a)}{w(e_{v_l})} \quad (13)$$

$$c_{v_l}^+(w(e_{v_l}), w_a) = c_{v_l}(w_a) + l_{v_l} w(e_{v_l}) c_0. \quad (14)$$

Equation (13) describes a delay surface which is strictly decreasing along $w(e_{v_l})$ direction($\partial d_{v_l}^+/\partial w(e_{v_l}) < 0$), and is a convex function along w_a direction($\partial^2 d_{v_l}^+/\partial w_a^2 > 0$). The maximum delay could be at two corners as specified in (15). The minimum delay could be at two corners or the minimum of the convex function, if the minimum is within the defined rectangle, as specified in (16).

$$d_{v_l,max}^+ = max(d_{v_l}^+(w_m, w_m(e_{vll})), d_{v_l}^+(w_m, w_M(e_{vll}))), \quad (15)$$

$$d_{v_l,min}^+ = min\left(d_{v_l}^+(w_M, w_m(e_{vll})), d_{v_l}^+(w_M, w_M(e_{vll})), \left\{d_{v_l}^+(w_M, w_a) \;\middle|\; \begin{array}{l} \frac{\mathrm{d}d_{v_l}^+(w_M, w_a)}{\mathrm{d}w_a} = 0 \\ w_m(e_{vll}) \le w_a \le w_M(e_{vll}) \end{array}\right\}\right). \quad (16)$$

Equation (16) would lead to a fourth-order equation that can be solved using iterative equation solvers such as GNU Scientific Library. The delay range of $\Omega_{v_r}^+$ can be calculated in the same way, or by (2)(5) if v_r is a leaf node.

Level-2 Nodes with Two Level-1 Children

For a level-2 node with two level-1 children, it is difficult to merge the branch DC regions described by complex equations with the $\curlywedge$ operator. We choose to use line sampling technique to overcome the problem. By taking delay samples from $\{d_{v_l}^+ \cap d_{v_r}^+\}$ and calculating the range of $c_{v_l}^+$ and $c_{v_r}^+$ for each sample, Ω_v can be obtained by merging sampled $\Omega_{v_l}^+$ and $\Omega_{v_r}^+$.

The maximum and minimum of d_v in Ω_v, $d_{v,min}$ and $d_{v,max}$, can be obtained from 3.1.3. Assuming that the set of p delay samples between $d_{v,min}$ and $d_{v,max}$ is $D_v = \{d_v^i, i = 1...p\}$, and substituting d_v^i back to (13), we have

$$w(e_{v_l})^i = \frac{l_{v_l} r_0 c_{vll}(w_a)}{d_v^i - d_{vll}(w_a) - \frac{l_{v_l}^2 r_0 c_0}{2}}. \quad (17)$$

Combined with (4) (14), the range of $c_{v_l}^{+i}$ can be derived. Again, we apply sampling technique on w_a because the range is difficult to solve analytically.

Assuming that the set of q wire width samples between $w_m(e_{vll})$ and $w_M(e_{vll})$ is $W_v = \{w_{aj}, j = 1...q\}$, for each delay d_v^i, we substitute w_{aj} back to (13) and get

$$w(e_{v_l})_j^i = \frac{l_{v_l} r_0 c_{vll}(w_{aj})}{d_v^i - d_{vll}(w_{aj}) - \frac{l_{v_l}^2 r_0 c_0}{2}}. \quad (18)$$

Substitute w_{aj} and $w(e_{v_l})_j^i$ into (14), we have

$$c_{v_l j}^{+i} = c_{vll(w_{aj})} + l_{v_l} w(e_{v_l})_j^i c_0. \quad (19)$$

The intervals $c_{v_l j}^{+i}$ sweep through give the range of $c_{v_l}^{+i}$. If we sample the delay evenly, the *sampled DC region* can be written as

$$\Omega_{vs} = \{(d_v^i, c_v^i), i = 1...p\},$$
$$where\ d_v^i = d_{v,min} + \frac{(i-1)}{(p-1)}(d_{v,max} - d_{v,min}) \quad (20)$$
$$c_{v_l,min}^{+i} + c_{v_r,min}^{+i} \le c_v^i \le c_{v_l,max}^{+i} + c_{v_r,max}^{+i}.$$

There might be more than one range for $c_{v_l}^{+i}$ and $c_{v_r}^{+i}$, and all combinations should be considered. However, the ranges will quickly merge together as the DC regions propagate toward the root node.

Level-2 Nodes with One Leaf Child

For level-2 nodes with level-1 child on the left and leaf child on the right, $\Omega^+_{v_l}$ and $\Omega^+_{v_r}$ can be obtained by the method in 3.1.3, (2) (5) respectedly. The merging procedure remains the same, except that $c^{+i}_{v_r}$ can be solved directly.

3.1.4 *Level-3 and Above Nodes*

The merging process described in 3.1.3 can also be applied to level-3 and above nodes. Assuming that the DC region of the left child is in sampled form. By combining (1) and (20), we get

$$\begin{aligned} d^{+k}_{v_l} &= d^k_{v_l} + \frac{l^2_{v_l} r_0 c_0}{2} + \frac{l_{v_l} r_0 c^k_{v_l}}{w(e_{v_l})}, \quad k = 1...p' \quad (21) \\ c^{+k}_{v_l} &= c^k_{v_l} + l_{v_l} w(e_{v_l}) c_0, \quad k = 1...p', \quad (22) \end{aligned}$$

where $w_m \leq w(e_{v_l}) \leq w_M$ and p' is the number of delay samples in $\Omega_{v_l s}$. Thus,

$$\begin{aligned} d^+_{v_l,max} &= max\left(\left\{ d^{+k}_{v_l} \middle| \begin{array}{l} w(e_{v_l}) = w_m \\ k = 1...p' \end{array} \right\}\right) \quad (23) \\ d^+_{v_l,min} &= min\left(\left\{ d^{+k}_{v_l} \middle| \begin{array}{l} w(e_{v_l}) = w_M \\ k = 1...p' \end{array} \right\}\right), \quad (24) \end{aligned}$$

and $d^+_{v_r,max}$, $d^+_{v_r,min}$ can be obtained in the same way. Taking a set of sampled delay, $D_v = \{d^i_v, i = 1...p\}$, from $\{d^+_{v_l} \cap d^+_{v_r}\}$ and substituting d^i_v into (21), we have

$$w(e_{v_l})^i_k = \frac{l_{v_l} r_0 c^k_{v_l}}{d^i_v - d^k_{v_l} - \frac{l^2_{v_l} r_0 c_0}{2}}, \quad k = 1...p'. \quad (25)$$

Substituting (25) into (22), we get

$$\begin{aligned} c^{+i}_{v_l k} &= c^k_{v_l} + \frac{l^2_{v_l} r_0 c_0 c^k_{v_l}}{d^i_v - d^k_{v_l} - \frac{l^2_{v_l} r_0 c_0}{2}} \\ &= \frac{(d^i_v - d^k_{v_l} + \frac{l^2_{v_l} r_0 c_0}{2})}{(d^i_v - d^k_{v_l} - \frac{l^2_{v_l} r_0 c_0}{2})} c^k_{v_l}, \quad k = 1...p'. \quad (26) \end{aligned}$$

Thus $c^{+i}_{v_l}$ can be calculated by overlapping the p' intervals from (26), and $\Omega^+_{v_l s} = \{(d^{+i}_{v_l}, c^{+i}_{v_l}), i = 1...p\}$ is obtained. Using the same way to obtain $\Omega^+_{v_r s}$, $\Omega_{vs} = \Omega^+_{v_l s} \curlywedge \Omega^+_{v_r s}$ can be obtained.

3.2 Top-Down Phase

The top-down phase is straight-forward. We first select a pair of target delay and capacitance load values (d_t, c_t) from Ω_v. Since a point in Ω_v represents at least one embedding, it is guaranteed to find an embedding that achieves the specified target delay and capacitance load. The capacitance load is further divided into c_{tl} and c_{tr} such that $c_{tl} + c_{tr} = c_t$, $(d_t, c_{tl}) \subset \Omega^+_{v_l}$, and $(d_t, c_{tr}) \subset \Omega^+_{v_r}$. If v_l is a leaf node, then $w(e_{v_l})$ is determined by (1). If v_l is a level-1 node, the feasible range of $w(e_{v_l})$ can be obtained by (13). If v_l is a level-2 or above node, then the DC region of v_l is in sampled form. For each sample in $\Omega_{v_l s}$, the range of $w(e_{v_l})$ can be obtained by (1). Once $w(e_{v_l})$ is chosen and the target delay and capacitance load of Ω_{v_l} are determined, we can proceed to determine the wire widths in T_{v_l}. Same approach applies to v_r. *ClockTune_Embed()* is given in Algorithm 3.

Algorithm 3 *ClockTune_Embed*(d_t, c_t, T_v)

```
Input: a clock tree $T_v$ with given routing rooted at node $v$
Output: an embedding of $T_v$
if $v$ is the root node then
  choose $(d_t, c_t)$ from $\Omega_v$
end if
choose $c_{tl}$ and $c_{tr}$ such that $c_{tl} + c_{tr} = c_t$, $(d_t, c_{tl}) \subset \Omega^+_{v_l}$, and $(d_t, c_{tr}) \subset \Omega^+_{v_r}$
foreach child node $u \in \{v_l, v_r\}$
    switch $u$
        case leaf node
            solve $w(e_u)$ by (1)
        case level-1 node
            solve the range of $w(e_u)$ by (13)
            choose $w(e_u)$ and calculate $(d_u, c_u)$
            call ClockTune_Embed$(d_u, c_u, T_u)$
        case level-2 or above node
            solve the range of $w(e_u)$ for each sample by (1)
            choose $w(e_u)$ and calculate $(d_u, c_u)$
            call ClockTune_Embed$(d_u, c_u, T_u)$
    end switch
end for
```

4. OPTIMALITY AND COMPLEXITY

We use sampling techniques to calculate DC regions, and the embeddings that do not map on the sampled delay will not be captured. However, the *sampling error* can be made arbitrarily small as we increase the sampling rate. The complexity of our algorithm is $O(max(p, q)pn)$, where n is the number of nodes, and p, q are the numbers of delay and wire width samples.

4.1 Optimality Analysis

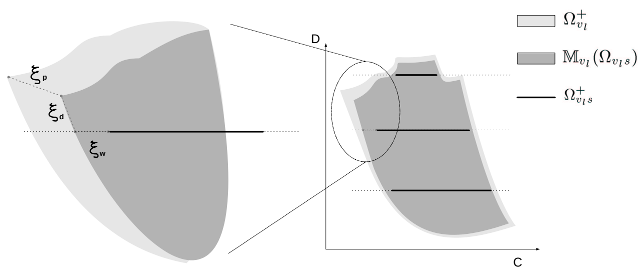


Figure 4: Illustration of sampling error components

To analyze the tradeoff between sampling rate and optimality, we first define the *distance* on the D-C plane and sampling error, and introduce two properties of errors. We further analyze the error on different level of nodes and prove the ϵ-optimality of *ClockTune*.

Definition 4. *Distance to a DC region*
The distance from a point $\mathbf{a}$ *on the D-C plane to a DC region* Ω_v *is defined as*
$dis(\mathbf{a}, \Omega_v) = min(\{\, |\mathbf{a} - \mathbf{b}| |\, \mathbf{b} \in \Omega_v\})$.

Definition 5. *Sampling Error*
The error of a sampled DC region Ω_{vs} *is defined as*
$\xi(\Omega_{vs}) = max(\{\, dis(\mathbf{a}, \Omega_{vs}) |\, \mathbf{a} \subset \Omega_v\})$.

The sources of the error of Ω_{vs} are the delay sampling, wire width sampling, and the sampling error from lower level nodes. Using the triangle inequality, we can separate these three components of the sampling error. Figure 4 gives an example of the property.

Property 1. *Error Composition*
$\xi(\Omega^+_{v_l s}) \leq \xi_p(\Omega^+_{v_l s}) + \xi_d(\Omega^+_{v_l s}) + \xi_w(\Omega^+_{v_l s})$, where $\xi_p(\Omega^+_{v_l s})$ is the propagated error from $\xi(\Omega_{v_l s})$, $\xi_d(\Omega^+_{v_l s})$ is the delay sampling error, and $\xi_w(\Omega^+_{v_l s})$ is the additional wire width sampling error after delay sampling.

Ω_v is generated from $\Omega^+_{v_l}$ and $\Omega^+_{v_r}$ by the $\curlywedge$ operator, which does not introduce additional errors. Thus we have the following property.

Property 2. *Error Bound*
The error of a sampled DC region Ω_{vs} is bounded by $\xi(\Omega_{vs}) \leq \xi(\Omega^+_{v_l s}) + \xi(\Omega^+_{v_r s})$.

4.1.1 Error Propagation

The first source of error comes from lower level nodes. Rewrite $d^+_{v_l}$ and $c^+_{v_l}$ into matrix form:

$$\begin{bmatrix} d^+_{v_l} \\ c^+_{v_l} \end{bmatrix} = \begin{bmatrix} 1 & \frac{l_{v_l} r_0}{w(e_{v_l})} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_{v_l} \\ c_{v_l} \end{bmatrix} + \begin{bmatrix} \frac{l^2_{v_l} r_0 c_0}{2} \\ l_{v_l} w(e_{v_l}) c_0 \end{bmatrix}. \quad (27)$$

The spectral norm of the matrix is

$$\lambda_{v_l} = 1 + \frac{l_{v_l} r_0}{2 w_m} \left(\frac{l_{v_l} r_0}{w_m} + \sqrt{4 + \left(\frac{l_{v_l} r_0}{w_m}\right)^2} \right). \quad (28)$$

$\xi_p(\Omega^+_{v_l})$ being the error propagated from lower level DC regions, we have

$$\xi_p(\Omega^+_{v_l s}) \leq \lambda_{v_l} \xi(\Omega_{v_l s}). \quad (29)$$

The upper bound of $\xi_p(\Omega^+_{v_l s})$ is determined by the error in the lower level and λ_{v_l}. As the wire length increases, λ_{v_l} also increases.

4.1.2 Sampling Error of Level-2 DC Regions

The other sources of error come from the sampling process. By fixing $w_m(e_{v_{ll}}) \leq w_a \leq w_M(e_{v_{ll}})$, (13) (14) form a subset of $\Omega^+_{v_l}$. The difference between the maximum and minimum $d^+_{v_l}$ of a given w_a is lower bounded by $d_{rm} = l_{v_l} r_0 c_{v_l}(w_m(e_{v_{ll}}))(\frac{1}{w_m} - \frac{1}{w_M})$. If we make the delay sampling interval $\delta_d < d_{rm}$, at least one point in each subset of $\Omega^+_{v_l}$ is on $\Omega^+_{v_l s}$. Due to the convex property of subsets of $\Omega^+_{v_l}$, for every $(d_a, c_a) \subset \Omega^+_{v_l}$, we can find a $(d_b, c_b) \subset \Omega^+_{v_l s}$ such that $|d_b - d_a| < \delta_d$ and $|c_b - c_a| < \delta_c$, where

$$\begin{aligned} \delta_c &= max \left(\left\{ \delta_d \left| \frac{\partial c^+_{v_l}(w(e_{v_l}), w_a)/\partial w(e_{v_l})}{\partial d^+_{v_l}(w(e_{v_l}), w_a)/\partial w(e_{v_l})} \right| \;\middle|\; \begin{array}{l} w_m \leq w(e_{v_l}) \leq w_M \\ w_m(e_{v_{ll}}) \leq w_a \leq w_M(e_{v_{ll}}) \end{array} \right\} \right) \\ &= \delta_d \frac{c_0 w^2_M}{r_0 c_{v_l}(w_m(e_{v_{ll}}))}. \end{aligned} \quad (30)$$

Thus the delay sampling error is bounded by

$$\xi_d(\Omega^+_{v_l s}) \leq \delta_d \sqrt{1 + \left(\frac{c_0 w^2_M}{r_0 c_{v_l}(w_m(e_{v_{ll}}))}\right)^2} = \delta_d k_{dv_l}. \quad (31)$$

For delay sample d^i_v, substitute (17) back to (14), we have

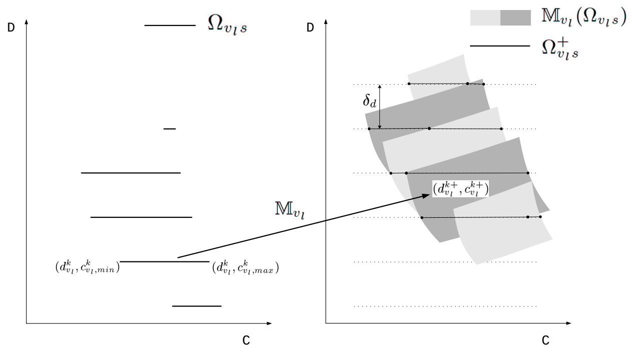
$$c^{+i}_{v_l}(w_a) = c_{v_l}(w_a) + l_{v_l} c_0 \frac{l_{v_l} r_0 c_{v_l}(w_a)}{d^i_v - d_{v_l}(w_a) - \frac{l^2_{v_l} r_0 c_0}{2}} \quad (32)$$

The error caused by wire width sampling with sampling interval δ_w is bounded by

$$\xi_w(\Omega^+_{v_l s}) \leq max \left(\left\{ \delta_w \left| \frac{\mathrm{d} c^{+i}_{v_l}(w_a)}{\mathrm{d} w_a} \right| \;\middle|\; \begin{array}{l} i = 1...p \\ w_m(e_{v_{ll}}) \leq w_a \leq w_M(e_{v_{ll}}) \end{array} \right\} \right) = \delta_w k_{wv_l}. \quad (33)$$

For a given clock routing, k_{wv_l} is a constant determined by $c_{v_{ll}}$, $c_{v_{lr}}$, $l_{v_{ll}}$, $l_{v_{lr}}$, and l_{v_l} only. For level-2 nodes, no error is propagated from either child, and $\xi(\Omega_{vs}) \leq \delta_d(k_{dv_l} + k_{dv_r}) + \delta_w(k_{wv_l} + k_{wv_r})$.

4.1.3 Sampling Error at Level-3 and above Nodes



D
$\Omega_{v_l s}$
$(d^k_{v_l}, c^k_{v_l,min})$
$(d^k_{v_l}, c^k_{v_l,max})$
$\mathbb{M}_{v_l}$
C
D
$\mathbb{M}_{v_l}(\Omega_{v_l s})$
$\Omega^+_{v_l s}$
δ_d
$(d^{k+}_{v_l}, c^{k+}_{v_l})$
C

Figure 5: The process of obtaining sampled DC-regions at level-3 and above nodes

The analyses of the error for level-2 nodes and the error propagation in a clock tree can be used to measure the sampling error at level-3 and above nodes. Figure 5 illustrates the process of obtaining $\Omega^+_{v_l s}$. From (21)(22), the slope of the upper and lower boundaries of $\{(d^+_{v_l k}, c^+_{v_l k})\}$ are $\frac{l_{v_l} r_0}{w_m}$ and $\frac{l_{v_l} r_0}{w_M}$. The minimum slope of left and right boundaries are $\frac{-r_0 c^k_{v_l,min}}{c_0 w^2_M}$ and $\frac{-r_0 c^k_{v_l,max}}{c_0 w^2_M}$. Thus The sampling error is bounded by the following inequality.

$$\begin{aligned} \xi_d(\Omega^+_{v_l s}) \leq\; & max \Bigg(\delta_d \sqrt{1 + \left(\frac{w_M}{l_{v_l} r_0}\right)^2}, \\ & \left\{ \delta_d \sqrt{1 + \left(\frac{c_0 w^2_M}{r_0 c^k_{v_l,min}}\right)^2} \;\middle|\; k = 1...p' \right\} \Bigg) = \delta_d k'_{dv_l} \end{aligned} \quad (34)$$

Again, the delay sampling error is proportional to the sampling interval times a constant related to the given routing. For level-3 and above nodes, $\xi_w(\Omega^+_{v_l s}) = 0$ because no wire width sampling is done. Thus $\xi(\Omega_{vs}) \leq \delta_d(k'_{dv_l} + k'_{dv_r}) + \lambda_{v_l}\xi(\Omega_{v_l s}) + \lambda_{v_r}\xi(\Omega_{v_r s})$. The upper bound of the sampling error at root node can be obtained by recursion, which lead to the following conclusion.

Theorem 1. *ε-Optimality*
The ClockTune Algorithm is ϵ-optimal.

Proof. The upper bound of the total sampling error of the DC region at root node v can be obtained by the following recursion inequality.

$$\begin{aligned} \xi(\Omega_{vs}) \leq\; & \delta_d(k_{dv_l} + k_{dv_l}) + \delta_w(k_{wv_l} + k_{wv_l}) \\ & + \lambda_{v_l}\xi(\Omega_{v_l s}) + \lambda_{v_r}\xi(\Omega_{v_r s}) \end{aligned} \quad (35)$$

By reducing the sampling intervals for delay and wire width at every node to one kth of the original values, the worst-case total sampling error can also be reduced to one kth of the value. The sampling error can be reduced to an arbitrarily small value; the algorithm is ϵ-optimal. $\square$

4.2 Complexity Analysis

Assuming a clock tree T_v has n nodes, and we always take p samples for delay and q samples for wire width. In the bottom-up phase, we need $O(1)$ time to construct the DC regions for leaf and level-1 nodes. Level-2 nodes require $O(pq)$ time for delay and wire width sampling. The other nodes need $O(p^2)$ time to combine at most p ranges for each delay sample. Thus, the complexity for the bottom-up phase is $O(max(p,q)pn)$. In the top-down phase, each wire width can be determined in $O(p)$ time, and the complexity is $O(pn)$. The overall runtime complexity is $O(max(p,q)pn)$. Since we only need to store the maximum and minimum values of the capacitance load of each delay sample, the memory requirement is $O(pn)$. From (35), the required sample numbers p,q in achieving the specified accuracy is determined by some polynomial of the input parameters, thus *ClockTune* takes pseudo-polynomial runtime and memory usage.

5. EXPERIMENTAL RESULTS

We implement our algorithm in C++ and run the program on a 128MB 533Mhz Pentium III PC. The benchmarks r1-r5 are taken from [9]. In this section, we investigate the runtime and optimality tradeoff and determine the sampling rate for reasonable accuracy. An example that demonstrates the incremental refinement capability of our algorithm is followed.

5.1 Optimality and Runtime Tradeoff

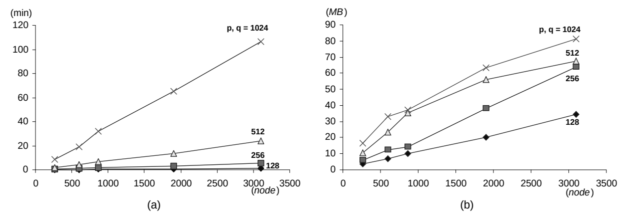


Figure 6: (a) Runtime and (b) memory usage with difference number of samples

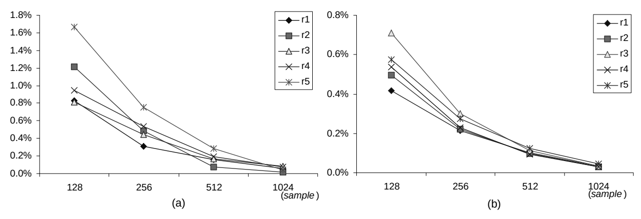


Figure 7: (a) Minimum delay and (b) minimum load error

We set p,q to $128, 256, 512, 1024$ in each run and record the minimum delay and capacitance load of the DC regions. All simulations use $r_0 = 0.03$, $c_0 = 2 \times 10^{-16}/\mu m^2$, and

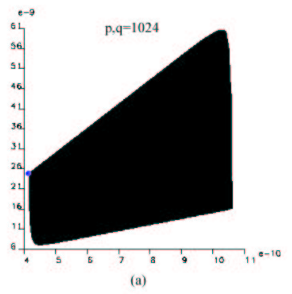


Figure 8: The DC regions of r5 with (a) $p,q = 1024$ (b) $p,q = 256$

acceptable wire width from $1\mu m$ to $4\mu m$. We choose the BB+DME [8] algorithm for routing generation, thus the capacitance load of the initial routing is the global optimal. The optimal delay is estimated by nonlinear curve fitting using (35). Figure 6 shows the runtime and memory usage, and Figure 7 shows relative errors between minimal and optimal delays and loads. The results show that *ClockTune* is very efficient. For r5, taking 256 samples is sufficient to find an embedding within 1% of the optimal solution and the runtime is less than 6 minutes. For 128 samples, the maximum error is less than 2% and the runtime is further reduced to 1.2 minutes while consuming only 34 MB of memory.

Figure 8(a) shows the DC region of r5 with $p,q = 256$, and Figure 8(b) with $p,q = 1024$. When $p,q = 256$, the errors are below 1%, thus p,q are fixed to 256 for the rest of the simulations. Note that all delay values are the Elmore delay values multiplied by $ln2$.

Table 2 lists the delay and capacitance load of initial routings and minimum delay/load embeddings by *ClockTune*. On average, *ClockTune* achieves 3.3X improvement on delay with only 22% increase on wire capacitance. All embeddings are verified by SPICE and the maximum skew is 12ps.

5.2 Incremental Refinement

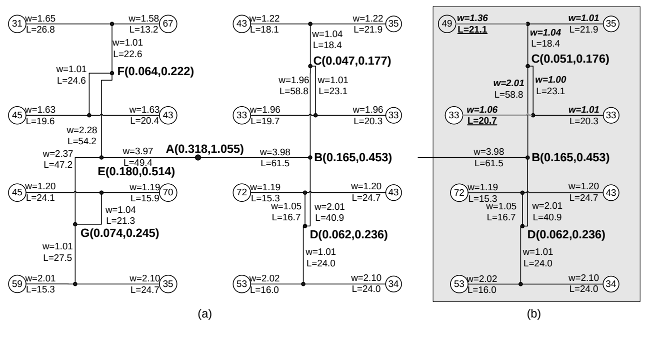


Figure 9: Routing examples (a) initial routing (b) incremental refinement for local changes

We will now proceed to show how incremental refinement is achieved by *ClockTune*. Figure 9(a) shows a clock routing with 16 sink nodes. The numbers in the circles are capacitance values in fF. Wire width and length are in μm. The *dc-pairs* followed by nodes A-G are the delay values in $10^{-12}s$ and capacitance load values in $10^{-12}F$ of the current embedding.

Input	Nodes	Initial($w = 1\mu m$)		*ClockTune*($1\mu m \leq w \leq 4\mu m$, *p=q=256*)							
		Delay	Wire Cap.	Opt. Delay	Min-delay embedding				Min-area embedding		CPU
		(ns)	(pF)	(Curve fitting)	Delay	Gain	Wire Cap.	Increase	Delay	Wire Cap.	(min.)
r1	267	1.097	30.79	0.362	0.364	3.0X	39.42	28.1%	1.090	30.91	0.40
r2	598	3.210	60.98	0.971	0.978	3.3X	75.41	23.7%	3.195	61.23	1.02
r3	862	4.590	79.13	1.374	1.381	3.3X	95.71	20.9%	4.506	79.58	1.66
r4	1903	13.18	161.3	3.806	3.828	3.4X	190.91	18.3%	13.13	162.0	3.37
r5	3101	24.88	242.5	6.945	7.010	3.5X	283.45	16.9%	24.74	243.9	5.65
average						3.3X		21.6%			

Table 2: Comparison of initial routings and minimum delay/load embeddings. The gains are measured by initial delays divided by minimum delays.

Figure 9(b) shows three changes in the upper-right portion of the routing; one of the sink capacitance and the lengths of two wire segments connecting to sink nodes are increased. We first re-construct the DC regions for the new clock routing. Since the dc-pair at node C does not reside in its new DC region, we could not find an embedding for T_C to achieve the same delay and capacitance load. However, the dc-pair at node B does fall in its new DC region, we are able to find an embedding for T_B that yields the same dc-pair while fixing the wire width of $\overline{BD}$. By choosing such an embedding, only $\overline{BC}$ and the wire segments in T_C need to be re-sized, and the rest of the clock routing were left intact.

If the designer chooses an optimal or near optimal embedding for the clock routing in the first place, the dc-pairs of the original embedding are more likely to fall out of the new DC regions as design changes occur. The changes might result in re-sizing many wire segments in order to re-balance the clock routing. In these cases, it might be desirable to choose a sub-optimal embedding for a subtree of the clock routing if future modifications are likely to occur. Our algorithm preserves the whole design space on the D-C plane and allows the designers to choose either optimal or sub-optimal embeddings depending on design requirements.

6. CONCLUSION AND FUTURE WORK

We present an ϵ-optimal zero-skew clock tree wire-sizing algorithm, *ClockTune*. The algorithm is guaranteed to find an ϵ-optimal embedding for the target delay or capacitance load requirement. For acceptable wire width ranging from $1\mu m$ to $4\mu m$, the algorithm achieves 3.3X improvement in delay with only 22% increase in wire area over the BB+DME algorithm. The algorithm is also capable of re-balancing the clock routing by local refinement should design changes occur. Moreover, the algorithm only takes pseudo-polynomial runtime and memory usage.

We plan to incorporate wire-sizing and buffer-insertion simultaneously into our algorithm framework. We will also investigate the inductance effect and adopt more accurate delay models in our future work.

7. REFERENCES

[1] Jason Cong and Majid Sarrafzadeh. Incremental physical design. In *Proceedings of the international symposium on Physical design, 2000*, pages 84–92. ACM Press, 2000.

[2] Joe G. Xi and Wayne W.-M. Dai. Useful-skew clock routing with gate sizing for low power design. In *Proceedings of the 33rd annual conference on Design automation conference*, pages 383–388. ACM Press, 1996.

[3] John Lillis, Chung-Kuan Cheng, and Ting-Ting Y. Lin. Optimal wire sizing and buffer insertion for low power and a generalized delay model. In *Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design*, pages 138–143. IEEE Computer Society Press, 1995.

[4] Rony Kay, Gennady Bucheuv, and Lawrence T. Pileggi. Ewa: exact wiring-sizing algorithm. In *Proceedings of the 1997 international symposium on Physical design*, pages 178–185. ACM Press, 1997.

[5] Chung-Ping Chen, Chris C. N. Chu, and D. F. Wong. Fast and exact simultaneous gate and wire sizing by lagrangian relaxation. In *Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, pages 617–624. ACM Press, 1998.

[6] Zhaoyun Xing and Prithviraj Banerjee. A parallel algorithm for zero skew clock tree routing. In *Proceedings of the 1998 international symposium on Physical design*, pages 118–123. ACM Press, 1998.

[7] I-Min Liu, Tan-Li Chou, Adnan Aziz, and D. F. Wong. Zero-skew clock tree construction by simultaneous routing, wire sizing and buffer insertion. In *Proceedings of the international symposium on Physical design, 2000*, pages 33–38. ACM Press, 2000.

[8] Ting-Hai Chao, Yu-Chin Hsu, Jan-Ming Ho, and A.B. Kahng. Zero skew clock routing with minimum wirelength. *Circuits and Systems II: Analog and Digital Signal Processing*, Volumn 39, Issue 11:799–814, Nov. 1992.

[9] R.-S. Tsay. Exact zero skew. In *Proceedings of the 1991 IEEE international conference on Computer-aided design*, pages 336–339, 1991.

[10] Lawrence T. Pillage and Ronald A. Rohrer. Asymptotic waveform evaluation for timing analysis. *Computer-Aided Design, IEEE Trans. on*, Volumn 9, NO. 4:352–366, Apr. 1990.

[11] Yu-Min Lee, Hing Yin Lai, and Charlie Chung-Ping Chen. Optimal spacing and capacitance padding for general clock structures. In *Proceedings of the conference on Asia South Pacific Design Automation Conference*, pages 115–119. ACM Press, 2001.

[12] Kazunori Inoue, Wataru Takahashi, Atsushi Takahashi, and Yoji Kajitani. Schedule-clock-tree routing for semi-synchronous circuits. *Fundamentals of Electronics, Communications and Computer Sciences, IEICE Trans. on*, Volumn E82-A, NO. 11:2431–2439, Nov. 1999.

[13] http://vlsi.ece.wisc.edu/Tools.htm