# Minimum-Delay/Power Zero-Skew Clock-Tree Optimization with Simultaneous Buffer-Insertion/Sizing and Wire-Sizing

Jeng-Liang Tsai, Tsung-Hao, Charlie Chung-Ping Chen
{ jltsai, tchen }@cae.wisc.edu, chen@engr.wisc.edu
Electrical and Computer Engineering, University of Wisconsin-Madison

## ABSTRACT

In 21st-Century VLSI design, clocking plays crucial roles for both performance and timing convergence. Minimum-delay/power zero-skew buffer-insertion/sizing and wire-sizing problems have long been considered intractable due to their non-convex nature. In this paper, we present *ClockTune*, a simultaneous buffer-insertion/sizing and wire-sizing algorithm which guarantees zero-skew and minimizes delay or power in pseudo-polynomial time. Extensive experimental results show that our algorithm executes very efficiently. For example, *ClockTune* achieves 40X delay improvement for buffering and sizing an industrial clock-tree with 3101 sink nodes on a 1.2GHz Pentium IV PC in 12 minutes compared with the initinal routing. 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.

## 1. INTRODUCTION

In the multi-giga-Hertz design era, clock design plays a crucial role in determining chip performance and facilitating timing and design convergence. First, clock-skew directly affects chip performance in a close to 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 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 the physical layout. From the above analysis, it is crucial to develop clock tuning algorithms that can balance clock-skew with minimum adjustments.

An excellent survey of interconnect optimization techniques can be found in [3]. Among the techniques suitable for clock-tree optimization are buffer-insertion/sizing and wire-sizing since these do not need to modify the existing routing. In [4] a three-stage optimization algorithm is proposed to minimize the delay and skew of a clock-tree. A reported 27X delay improvement was reported by buffer-insertion and buffer-sizing. In [5] an iterative algorithm performs wire-sizing one segment at a time and about 1.5X to 3X improvement on minimum delay was observed. Two major approaches have been used to integrate buffer insertion/sizing and wire-sizing techniques for delay and power optimization. In [6] [7] [8] the simultaneous buffer-insertion/sizing and wire-sizing problems are formulated as optimization problems, in which the maximum delay of each sink node is constrained. In [9] [10] bottom-up dynamic programming algorithms based on the method in [11] are used to find optimal solutions for a subtree and propagate the solutions up toward the root node. These methods perform optimizations without modifying the clock-routing, but do not guarantee zero-skew.

Recent work [12] integrates wire-sizing into the Deferred-Merging Embedding (DME) algorithm [13] which allows a zero-skew clock-tree to benefit from wire-sizing and buffer-insertion. However, the zero-skew property is achieved by moving the merging points and the clock-routing might be changed to accommodate the skew caused by design changes. This may affect the detail routing. To the best of authors' knowledge, there is no existing simultaneous wire-sizing and buffer-insertion/sizing algorithm which finds the minimum delay or power zero-skew solutions without modifying the existing routing.

In this paper, we propose a novel clock-tuning algorithm, *ClockTune*, which considers buffer-insertion/sizing and wire-sizing at the same time while maintaining the clock-tree zero-skew. *ClockTune* first calculates the feasible delay and power information for each node in a bottom-up fashion. After the desired delay and power is chosen from the feasible region, a buffering and wire-sizing is determined in a top-down fashion. Although we focus on achieving zero-skew, *ClockTune* can also be used to achieve useful skew to tackle timing problems. Moreover, if the clock-routing encounters design changes, *ClockTune* is able to re-balance the clock-tree by local adjustment.

The rest of this paper is organized as follows: in Section 2 we formulate the problems and introduce the models and notations we use in this work; in Section 3 the fundamentals of our algorithm are introduced; Section 4 provides the algorithm framework and gives the details of our *ClockTune* algorithm; Section 5 is the complexity analyses, and Section 6 is our experimental results.

## 2. PRELIMINARIES

The minimum-delay/power zero-skew wire-sizing(min-ZSWS) problem was solved in [14], and the proposed method provides a good basis for understanding this work. However, [14] did not consider buffer-insertion/sizing which is a more effective way of reducing clock delay. We first define both problems and repeat part of the conclusions of [14] to make this work self-contained.

**Problem Definition** 1. *Minimum-Delay/Power Zero-Skew Wire-Sizing(min-ZSWS) Problem :*
*Given a clock-tree $T$, find a set of feasible wire-widths with bounded delay and power consumption such that the zero-skew constraint is satisfied and the delay or power is minimized.*

**Problem Definition** 2. *Minimum-Delay/Power Zero-Skew Buffer-Insertion/Sizing and Wire-Sizing(min-ZSBWS) Problem :*
*Given a clock-tree $T$, find a set of feasible buffer locations, buffer widths, and wire-widths with bounded delay and power such that the zero-skew constraint is satisfied and the delay or power is minimized.*

### 2.1 Notations

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. In this work, buffers are only allowed to be inserted right above a node and no buffer is allowed to insert above a leaf node.

| | |
|---|---|
| $T_v$ | 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 $e_v$ |
| $b_v$ | The buffer above node $v$, $b_v = \phi$ if not buffered |
| $w(e_v)$ | The wire-width of edge $e_v$ |
| $w(b_v)$ | The output resistance of buffer $b_v$, $w(\phi) = 0$ |
| $r(e_v)$ | The resistance of wire $e_v$, $r(e_v) = \frac{l_v r_0}{w(e_v)}$ |
| $c(e_v)$ | The capacitance of wire $e_v$, $c(e_v) = l_v w(e_v) c_0$ |
| $d_v$ | Elmore-delay from node $v$ to any leaf node in $T_v$ |
| $c_v$ | Total down-stream capacitance seen at node $v$ |
| $c_{sv}$ | Total capacitance shielded from node $v$ by buffers |
| $r_0, c_0$ | Wire resistance and capacitance per $\mu m^2$ |
| $r_b, c_b$ | Unit-size gate resistance and capacitance |
| $w_m, w_M$ | Predefined minimum and maximum wire-width |
| $w_{bm}, w_{bM}$ | Predefined Minimum and maximum buffer-size |

**Table 1: Notations in this work**

### 2.2 Delay and Power Models

There are two delay components in a clock-tree: interconnect-delay and gate-delay. Although our *ClockTune* algorithm makes no presumption on the delay model, using the resistance-capacitance(RC) models for interconnects and buffers and the Elmore-delay model for delay calculation reduces the complexity of our algorithm. For a wire with length $l$ and width $w$, the wire resistance is $\frac{lr_0}{w}$ and the wire capacitance is $lc_0w$. The wire capacitance is further divided into two equal capacitors attached at both ends of the wire. For a buffer with gate width $w_b$, the gate capacitance at its input is $w_b c_b$. The gate is modeled as a ramp voltage source with an effective output resistance $\frac{r_b}{w_b}$. The ramp voltage source has a delay $t_c$, which models the intrinsic delay of the buffer.

The power consumed by the clock-tree can be modeled as $P = fCV^2 + P_s + P_l$, where $f$ is the switching frequency; $V$ is the voltage swing; and $C$ is the total interconnect capacitance, gate capacitance of the buffers, and sink loads. $P_s$ accounts for the buffer short-circuit power and $P_l$ accounts for the leakage power. In a usual design, the last two terms are usually much smaller and the total capacitance is a good measure of the total power consumption [15].

## 3. DESIGN-SPACE AND DC REGION

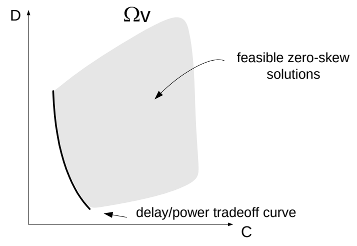

**Figure 1: The projection of feasible design solutions on a 2-dimensional plane**

Considering the min-ZSWS problem, if $T_v$ has $n$ edges then there are $n$ wire-widths to be determined. Every *embedding* of $T_v$ (a set of wire-widths which satisfies zero-skew and wire-width constraints) is a point in the $n$-dimensional design-space. Since we are only interested in the delay and power of the embeddings, we can project all the embeddings onto the *D-C plane*: the X-Y plane with delay value on Y-axis and capacitance load value on X-axis. The projection of the embeddings form a *DC region*, $\Omega_v$, on the *D-C plane*, is shown in Figure 1. The lower-left edge of the *DC region* is the delay/power tradeoff curve, and previous works [9] [10] [11] have emphasized finding the solutions which lie on this curve while pruning-out sub-optimal solutions. These approaches have two drawbacks. First, the combinations that lie on the curve grow superlinearly [11]. Second, early pruning sub-optimal solutions may result in sub-optimal global solutions due to the non-convex nature of the min-ZSWS problem.

In [14], a different approach is used to solve the min-ZSWS problem which relies on the following property:

**Property** 1. *Existence Property :*
*For every point $p_v = (d_v, c_v) \subset \Omega_v$, there exists at least one pair of points $p_{v_l} = (d_{v_l}, c_{v_l}) \subset \Omega_{v_l}$ and $p_{v_r} = (d_{v_r}, c_{v_r}) \subset \Omega_{v_r}$ such that the corresponding embeddings of $T_{v_l}$ and $T_{v_r}$ are the same as in the embedding of $T_v$ from $p_v$.*

The existence property is the restatement that for a feasible design of $T_v$, its design of subtrees $T_{v_l}$ and $T_{v_r}$ are also feasible, thus their projections are in $\Omega_{v_l}$ and $\Omega_{v_r}$. As illustrated in Figure 2, for a point $p_v$, at least one pair of $p_{v_l}$ and $p_{v_r}$ satisfies the following equations

$$c_v = \sum_{u \subset v_l, v_r} (c_u + l_u w(e_u) c_0) \quad (1)$$

$$d_v = d_u + \frac{l_u^2 r_0 c_0}{2} + \frac{l_u r_0}{w(e_u)}, u \subset v_l, v_r \quad (2)$$

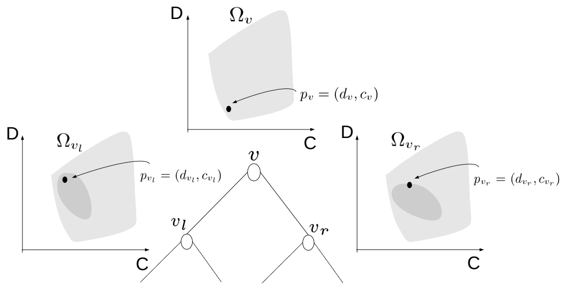

**Figure 2: Illustration of the existence property**

and $w(e_{v_l})$, $w(e_{v_r})$ can be calculated by (2).

It is worth to mention that in the Elmore-delay model a capacitor is used to model a RC tree and the calculated delay only matches the first moment of the exact impulse response. If a more accurate delay model is required, a resistor-capacitor (RC) model or capacitor-resistor-capacitor (CRC) model can be adopted to model a RC tree [16]. By adding another axis to the *D-C plane* for the additional parameter, it forms a *D-C space*. The *DC region* becomes the projection of the embeddings on the *D-C space*.

In the min-ZSBWS problem, buffering is allowed and $c_v$ is the total capacitance of $T_v$ minus the capacitance shielded by first-level buffers, $c_{sv}$, below $v$. Inserting a buffer also changes the signal polarity, thus $\Omega_v$ is split into two sets. $\Omega_{vp}$ is the projection of even-numbered buffered embeddings, $\Omega_{vn}$ is the projection of odd-numbered buffered embeddings, and $\Omega_v = \Omega_{vp} \cup \Omega_{vn}$. Property 1 still holds for the buffered case and (1) (2) can be rewritten as

$$c_v = \sum_{u \subset v_l, v_r} (c_u + l_u w(e_u) c_0 + w(b_u) c_b) \tag{3}$$

$$d_v = \begin{cases} d_u + \frac{l_u^2 r_0 c_0}{2} + \frac{l_u r_0}{w(e_u)} & , b_u = \phi \\ d_u + \frac{r_b c_u}{w(b_u)} + t_c + \frac{l_u^2 r_0 c_0}{2} + \frac{l_u r_0 c_b w(b_u)}{w(e_u)} & , otherwise \end{cases} \tag{4}$$

By Property 1, at least one set of buffering decision, buffer-widths, and wire-widths satisfies (3) (4) for a given set of $p_v$, $p_{v_l}$, and $p_{v_r}$. The feasible embeddings are actually implied in the *DC regions* and we can avoid handling the growing design combinations by storing the *DC regions* associated with every node. In the next section we will show how to obtain the *DC regions* and $p_v$, $p_{v_l}$, and $p_{v_r}$.

## 4. THE CLOCKTUNE ALGORITHM

We propose a dynamic programming algorithm, *ClockTune*, to solve the min-ZSWS and min-ZSBWS problems. *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 buffer locations, buffer-widths, and wire-widths. The framework of *ClockTune* is given as a pseudocode in Algorithm 1 and Figure 3 illustrates its procedures.

### 4.1 Zero-Skew Wire-Sizing Algorithm

In this subsection we detail the bottom-up and top-down process of *ClockTune* in solving the min-ZSWS problem.

#### 4.1.1 Bottom-up Phase

We first introduce the definition of *branch DC region* and the associated $\curlywedge$ operator to facilitate our discussion fol-

**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 4.1.1 for min-ZSWS and in 4.2 for min-ZSBWS */
  Choose $(d_t, c_t)$ from $\Omega_v$
  /* choose the minimum-delay or power solution for $T_v$ */
  Call ClockTune_Embed($d_t, c_t, T_v$)
  /* top-down sizing/buffer-insertion, detailed in 4.1.2 */
```

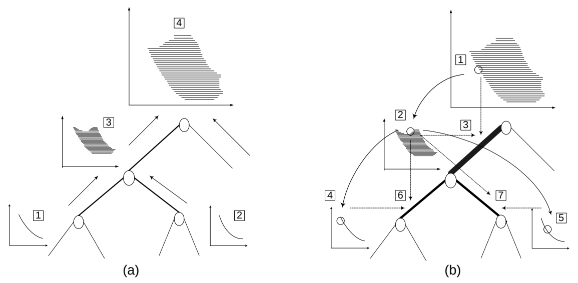

**Figure 3: Illustration of *ClockTune* (a) the bottom-up phase (b) the top-down phase**

lowed by introducing the wire-sizing transformation of calculating the *branch DC region*.

**Definition** 1. *Branch DC Region :*
*The branch DC region of node $v$, $\Omega_v^+ = \{(d_v^+, c_v^+)\}$, is the projection of all embeddings of $T_v^+ = \{e_v \cup T_v\}$ on the D-C plane.*

**Definition** 2. $\curlywedge$ *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:*

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

**Lemma** 1. *Wire-sizing transformation*
*$\Omega_v^+$ can be obtained from $\Omega_v$ by the transformation $\mathbb{W}_v : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ described as follows:*

$$d_v^+ = d_v + \frac{l_v r_0}{w(e_v)} \left( \frac{l_v w(e_v) c_0}{2} + c_v \right) \tag{5}$$

$$c_v^+ = c_v + l_v w(e_v) c_0, \tag{6}$$
$$where (d_v, c_v) \subset \Omega_v, w_m \le w(e_v) \le w_M$$

The *ClockTune_DC($T_v$)* subroutine of the *ClockTune* algorithm can now be written as Algorithm 2. For a leaf node $v$, $c_v$ is the load capacitance and hence a constant. To enforce the zero-skew constraint, we set $d_v$ to 0 for all leaf nodes, and $\Omega_v = \{(d_v, c_v)\}$ are single points on the D-C plane. Although our focus is on the min-ZSWS problem, *ClockTune* can accept arbitrary skew values by simply assigning different $d_v$ for each leaf nodes. *ClockTune* can also be extended to accept bounded-skew constraints where $\Omega_v$ of the leaf nodes become vertical segments.

For a level-1 node $v$, the closed-form solution of $\Omega_v$ can be obtained by solving (5) (6) for $v_l$, $v_r$, and impose the

**Algorithm 2** *ClockTune_DC($T_v$) in min-ZSWS problem*

```
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($T_{v_r}$)
  $\Omega_{v_l}^+ \leftarrow \mathbb{W}_{v_l}(\Omega_{v_l})$
  $\Omega_{v_r}^+ \leftarrow \mathbb{W}_{v_r}(\Omega_{v_r})$
  $\Omega_v \leftarrow \Omega_{v_l}^+ \curlywedge \Omega_{v_r}^+$
end if
```

zero-skew constraint $d_{v_l}^+ = d_{v_r}^+$. The solution is as below.

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

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

Equations (7) (8) represent a strictly decreasing curve on the *D-C plane*. However, the closed-form solution of $\Omega_v$ for a level-2 node $v$ is difficult to obtain. Sampling techniques are applied to sample and store $\Omega_{v_l}^+$ and $\Omega_{v_r}^+$ which are then combined into $\Omega_v$. We first take $p$ samples on the delay range $d_{v_l}^+ \cap d_{v_r}^+$, then take $q$ samples for $w(e_u)$ (assuming $u$ is the level-1 child of $v$). For each sample of $w(e_u)$, (7) (8) gives a single point, and a subset of $\Omega_u^+$ which is also a strictly decreasing curve can be obtained. The intersection points of these $q$ curves and $p$ delay samples can be calculated, and the ranges those points span can be captured. By taking the same $p$ delay samples on the other child node, $\Omega_v = \Omega_{v_l}^+ \curlywedge \Omega_{v_r}^+$ can be obtained. The procedure is illustrated in Figure 4. In a *sampled DC region*, each delay sample is associated with one or more capacitance ranges. The *branch DC region* of a *sampled DC region* can be solved by (5) (6) analytically and again we perform sampling on the delay to obtain the *sampled DC region* for level-3 and above nodes.

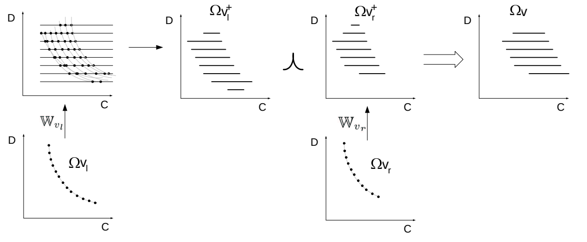

**Figure 4: Obtain the DC region of a level-2 node by sampling techniques**

### 4.1.2 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 $\Omega_v$, which can be the minimum-delay or minimum-power solution. The capacitance load $c_t$ 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 (2). If $v_l$ is a level-1 node, the feasible range of $w(e_{v_l})$ can be obtained by solving (5). If $v_l$ is a level-2 or above node, then the DC region of $v_l$ is in a sampled form. For each sample in $\Omega_{v_l}$, the range of $w(e_{v_l})$ can be obtained by solving (5) and at least one range of $w(e_{v_l})$ is feasible by Property 1. Once $w(e_{v_l})$ is chosen, the target delay and capacitance load of $\Omega_{v_l}$ are determined and we can proceed to determine the wire-widths in $T_{v_l}$. The same approach applies to $v_r$. *ClockTune_Embed()* is given in Algorithm 3.

**Algorithm 3** *ClockTune_Embed($d_t, c_t, T_v$) in min-ZSWS*

```
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 (2)
    case level-1 node
      solve the range of $w(e_u)$ by (5)
      choose a $w(e_u)$ 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 every sample in $\Omega_u$ by (5)
      choose a $w(e_u)$ and calculate $(d_u, c_u)$
      call ClockTune_Embed($d_u, c_u, T_u$)
  end switch
end for
```

### 4.1.3 $\epsilon$-Optimality

It has been shown that the *ClockTune* algorithm for the min-ZSWS problem is $\epsilon$-optimal; for any given $\epsilon$, *ClockTune* finds a solution that is within $\epsilon$ distance to the optimal delay/power solution by adjusting numbers of samples on delay and wire-width.

## 4.2 Zero-Skew Buffer-Insertion/Sizing and Wire-Sizing Algorithm

In the min-ZSBWS problem $\Omega_v$ is split into $\Omega_{vp}$ and $\Omega_{vn}$. When applying $\mathbb{W}_v$, the polarities of the *DC regions* remain unchanged. If a buffer is inserted, the total capacitance increases, but the capacitance seen by upstream nodes is reduced. Thus, we need to expand the *D-C plane* into the *D-C space* where $d_v$ is on the Y-axis, $c_{sv}$ is on the X-axis, and $c_v$ is on the Z-axis. We now define another transformation for buffer-insertion/sizing.

**Lemma** 2. *Buffer-insertion/sizing transformation : $\Omega_{vn}^+$ with buffer inserted above $v$ can be obtained from $\Omega_{vp}$ by the transformation $\mathbb{B}_v : \mathbb{R}^5 \rightarrow \mathbb{R}^3$ described as follows:*

$$d_{vn}^+ = d_{vp} + \frac{r_b c_{vp}}{w(b_v)} + t_c + \frac{l_v^2 r_0 c_0}{2} + \frac{l_v r_0 c_b w(b_v)}{w(e_v)} \quad (9)$$

$$c_{vn}^+ = c_b w(b_v) + l_v w(e_v) c_0, \quad (10)$$

$$c_{svn}^+ = c_{svp} + c_{vp} \quad (11)$$

$$\textit{where } (d_{vp}, c_{vp}, c_{svp}) \subset \Omega_{vp},$$
$$w_m \le w(e_v) \le w_M, w_{bm} \le w(b_v) \le w_{bM},$$
$$p,\ n \textit{ are interchangeable.} \quad (12)$$

To obtain the 3-dimensional *DC regions*, sampling is first performed on the delay and shielded capacitance values along

Y and X directions. To create the *branch DC regions* above a buffered node, we also take samples on $w(b_v)$. Note that although we take samples on two more variables, the sampling originally required on the child branch width can be eliminated by the fact that $c_{vp} = c^+_{svn} - c_{svp}$ from (11). Due to the space limit, we omit the details and illustrate the procedure in Figure 5 and Algorithm 4. The top-down algorithm is basically the same except that the power is estimated by $c_v + c_{sv}$ and both $c_t = c_{tl} + c_{tr}$ and $c_{st} = c_{stl} + c_{str}$ have to be satisfied when choosing $p_{vl}$ and $p_{vr}$.

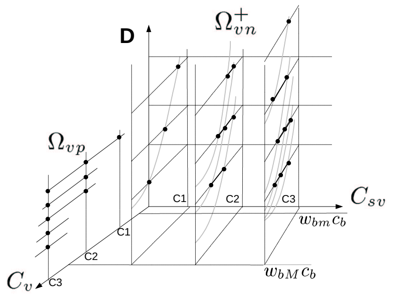

**Figure 5: Illustration of the procedure to generate the *branch DC region* above a buffered node**

**Algorithm 4** *ClockTune_DC($T_v$) in min-ZSBWS problem*

```
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
    $\Omega_{vp} \leftarrow (d_v, c_v, c_{sv}), \Omega_{vn} \leftarrow \phi$
else {$v$ is an internal node}
    call ClockTune_DC($T_{v_l}$)
    call ClockTune_DC($T_{v_r}$)
    $\Omega^+_{v_lp} \leftarrow \mathbb{W}_{v_l}(\Omega_{v_lp}) \cup \mathbb{B}_{v_l}(\Omega_{v_ln})$
    $\Omega^+_{v_ln} \leftarrow \mathbb{W}_{v_l}(\Omega_{v_ln}) \cup \mathbb{B}_{v_l}(\Omega_{v_lp})$
    $\Omega^+_{v_rp} \leftarrow \mathbb{W}_{v_r}(\Omega_{v_rp}) \cup \mathbb{B}_{v_r}(\Omega_{v_rn})$
    $\Omega^+_{v_rn} \leftarrow \mathbb{W}_{v_r}(\Omega_{v_rn}) \cup \mathbb{B}_{v_r}(\Omega_{v_rp})$
    $\Omega_v \leftarrow (\Omega^+_{v_lp} \curlywedge \Omega^+_{v_rp}) \cup (\Omega^+_{v_ln} \curlywedge \Omega^+_{v_rn})$
end if
```

## 4.3 Slew-rate Control

One of the purposes for buffer-insertion is to adjust the clock slew-rate. If the loading capacitance of a buffer is too large, the output signal will have a slow rise and fall time, and it in turn increases the short-circuit power of downstream buffers. One way to control the slew-rate is to limit the loading capacitance to a certain value such that the slew-rate of the buffer is bounded to the desired value. This constraint can be taken care of easily by limiting $c_v$ during the bottom-up phase. In this manner, it is guaranteed that the embeddings we get during top-down phase will not have any buffer driving a load that exceed the predefined upper limit. During the bottom-up phase, the *DC regions* might grow very large because of the ill-buffered embeddings. Again we can set upper limits on $d_v$ and $(c_{sv} + c_v)$. Since $c_v$ has been limited by the maximum buffer loading value, which is usually small, imposing the limit on $c_{sv}$ is sufficient. These limits are equivalent to adding 3 cutting planes in the *D-C space* and only consider the *DC regions* that lie inside the cube on the first-octant.

## 4.4 Incremental Refinement

When clock-routing undergoes design changes and the clock-tree is no longer zero-skew, *ClockTune* can be used to perform incremental refinement in the way that follows. First, the DC regions are reconstructed from affected nodes until it reaches node $v$ such that $T_v$ covers all design changes. If the projection of the original embedding of $T_v$, $(\tilde{d}_v, \tilde{c}_v, \tilde{c}_{sv})$, falls in the new *DC region*, we take this point and run *ClockTune_Embed($\tilde{d}_v, \tilde{c}_v, \tilde{c}_{sv}$)* to determine the buffering and wire-sizing of $T_v$. The rest of the clock-tree is not aware of these design changes because $(\tilde{d}_v, \tilde{c}_v, \tilde{c}_{sv})$ exposed to the rest of the clock-tree remains the same. Otherwise, we keep updating the *DC regions* toward the root node until the original projection falls in the new *DC region*.

## 5. COMPLEXITY

Assuming a clock-tree $T_v$ has $n$ nodes, we always take $p$ samples for delay and $q$ samples for wire-width. In the min-ZSWS problem, it takes $O(1)$ 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 $p$ 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 $O(pn)$. In the min-ZSBWS problem, we take $r$ samples on the shielded capacitance value and the straight forward implementation requires $O(p^2qr^2n)$ runtime. By exploring the properties of (9) (10) (11) the runtime can be reduced to $\sim O(pqr^2n)$, and the memory requirement is $O(prn)$.

## 6. EXPERIMENTAL RESULTS

We implement our algorithm in C++ and run the program on a 1GB 1.2GHz Pentium IV PC. The benchmarks r1-r5 are taken from [17]. All simulations use $r_0 = 0.03$, $c_0 = 2 \times 10^{-16}/\mu m^2$, $w_m = 0.3\mu m$, $w_M = 3\mu m$. The parameters of the buffers are $c_b = 40fF$, $r_b = 100\Omega$, $t_c = 30ps$, and $w_{bm} = 1$, $w_{bM} = 10$. The initial routings are generated by the BB+DME [13] algorithm. The numbers of samples used in the min-ZSWS problem are $p = q = 256$. The numbers of samples used in the min-ZSBWS problem are $p = q = r = 64$.

Table 2 shows the minimum-delay and minimum-power solutions for the min-ZSWS problem. If the initial routing does not use the minimum wire-width, then both the delay and power can be lowered by performing wire-sizing. Table 3 shows the minimum-delay and minimum-power solutions for the min-ZSBWS problem. The delay is dramatically lower than that of the initial routing even for the minimum-power solution, and the minimum-delay solutions have more than 2X speedup compared with the minimum-power solution. Note that the delays shown in the figures and tables are the Elmore-delays multiplied by $ln2$. Figure 6 shows the *DC regions* of the root node in r5 for the min-ZSWS and min-ZSBWS problems.

## 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.

| Input | Initial($w = 1\mu m$) | | *ClockTune*($w_m = 0.3\mu m, w_M = 3\mu m, p = q = 256$) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | delay | load | minimum-delay solution | | | | minimum-power solution | | | | CPU |
| (#nodes) | (ns) | (pF) | Delay | Gain | Load | Gain | Delay | Gain | Load | Gain | (min.) |
| r1(267) | 1.009 | 45.2 | 0.306 | 3.59 | 34.5 | 1.56 | 1.905 | 0.58 | 24.7 | 2.18 | 0.34 |
| r2(598) | 3.210 | 93.6 | 0.762 | 4.21 | 67.7 | 1.59 | 5.195 | 0.62 | 53.0 | 2.04 | 0.83 |
| r3(862) | 4.590 | 126.7 | 1.152 | 3.99 | 92.5 | 1.55 | 7.838 | 0.59 | 73.6 | 1.95 | 1.28 |
| r4(1903) | 13.184 | 266.2 | 3.288 | 4.01 | 184.3 | 1.61 | 24.802 | 0.53 | 156.8 | 1.89 | 2.61 |
| r5(3101) | 24.883 | 413.0 | 6.093 | 4.08 | 286.6 | 1.58 | 47.958 | 0.52 | 248.1 | 1.83 | 4.35 |

**Table 2: The delay and power before and after wire-sizing. The gains are measured by the initial values divided by the optimized values.**

| Input | Initial($w = 1\mu m$) | | *ClockTune*($w_m = 0.3\mu m, w_M = 3\mu m, w_{bm} = 1, w_{bM} = 10, p = q = r = 64$) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | delay | load | minimum-delay solution | | | | | minimum-power solution | | | | | CPU |
| | (ns) | (pF) | Delay | Gain | Load | Gain | Buffers | Delay | Gain | Load | Gain | Buffers | (min.) |
| r1(267) | 1.009 | 45.2 | 0.145 | 7.54 | 36.5 | 1.24 | 38 | 0.382 | 2.875 | 28.1 | 1.61 | 23 | 1.1 |
| r2(598) | 3.210 | 93.6 | 0.200 | 16.06 | 88.7 | 1.06 | 80 | 0.636 | 5.05 | 64.6 | 1.45 | 66 | 2.7 |
| r3(862) | 4.590 | 126.7 | 0.236 | 19.43 | 108.0 | 1.17 | 119 | 0.563 | 8.15 | 85.1 | 1.49 | 72 | 3.0 |
| r4(1903) | 13.184 | 266.2 | 0.345 | 38.19 | 229.7 | 1.16 | 251 | 0.927 | 14.22 | 193.8 | 1.37 | 148 | 7.7 |
| r5(3101) | 24.883 | 413.0 | 0.563 | 44.20 | 324.9 | 1.27 | 342 | 1.036 | 24.03 | 323.7 | 1.28 | 281 | 11.2 |

**Table 3: The delay and power before and after buffer-insertion/sizing and wire-sizing.**

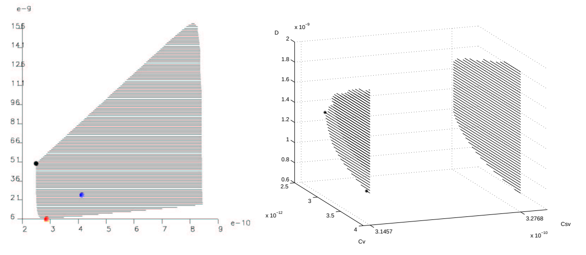

**Figure 6: The DC regions of the root node of $r5$ in (left) min-ZSWS and (right) min-ZSBWS problems. The marks indicate the minimum-delay and minimum-power solutions**

[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] Jason Cong, Zhigang Pan, Lei He, Cheng-Kok Koh, and Kei-Yong Khoo. Interconnect design for deep submicron ics. In *Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*, pages 478–485. IEEE Computer Society, 1997.

[4] X. Zeng, D. Zhou, and Wei Li. Buffer insertion for clock delay and skew minimization. In *Proceedings of the 1999 international symposium on Physical design*, pages 36–41. ACM Press, 1999.

[5] Sachin S. Sapatnekar. Rc interconnect optimization under the elmore delay model. In *Proceedings of the 31st annual conference on Design automation conference*, pages 387–391. ACM Press, 1994.

[6] 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.

[7] J. Cong, C. Koh, and K. Leung. Simultaneous buffer and wire sizing for performance and power optimization. In *Proceedings of the 1996 international symposium on Low power electronics and design*, pages 271–276. IEEE Press, 1996.

[8] 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.

[9] Takumi Okamoto and Jason Cong. Buffered steiner tree construction with wire sizing for interconnect layout optimization. In *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, pages 44–49. IEEE Computer Society Press, 1996.

[10] Charles J. Alpert, Anirudh Devgan, and Stephen T. Quay. Buffer insertion with accurate gate and interconnect delay computation. In *Proceedings of the 36th ACM/IEEE conference on Design automation conference*, pages 479–484. ACM Press, 1999.

[11] L.P.P.P. van Ginneken. Buffer placement in distributed rc-tree networks for minimal elmore delay. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 865–868, 1990.

[12] 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.

[13] 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.

[14] The authors. Epsilon-optimal min-delay/area zero-skew clock tree wire-sizing in pseudo-polynomial time. In *to be appeared in the international symposium on Physical design, 2003.* ACM Press, 2003.

[15] Satyamurthy Pullela, Noel Menezes, Junaid Omar, and Lawrence T. Pillage. Skew and delay optimization for reliable buffered clock trees. In *Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design*, pages 556–562, 1993.

[16] Peter R. O'Brien and Thomas L. Savarino. Modeling the driving-point characteristic of resistive interconnect for accurate delay estimation. In *Proceedings of the 1989 IEEE/ACM international conference on Computer-aided design*, pages 512–515, 1989.

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