Buffer Insertion for Clock Delay and Skew Minimization*

X. Zeng[1, 2], D. Zhou[1], Wei Li[1]

[1]Department of Electrical and Computer Engineering
University of North Carolina at Charlotte, NC 28223

[2]Department of Electronic Engineering
Fudan University, Shanghai 200433, China

***Abstract* — Buffer insertion is an effective approach to achieve both minimal clock signal delay and skew in high speed VLSI circuit design. In this paper, we develop an optimal buffer insertion and sizing scheme. Particularly, due to the buffer to buffer delay is a convex function of buffer positions in a clock tree, we show that the minimal clock delay can be obtained by equalizing derivatives of this convex function, and the minimal skew can be obtained by equalizing delay functions of different source to sink paths. Based on this theory, we further develop a three-stage method to initially insert buffers in a given clock routing tree, minimize delay by optimizing buffer positions, and minimize skew by buffer level augment and buffer size refinement. The presented algorithm achieves both minimal delay and skew in real clock tree design.**

1. Introduction

In the coming century, the clock rate of microprocessors will approach multi-GHz. This trend will last as process technology moves into the deep submicron level. The drastically increased requirement for high performance and high speed VLSI circuits has posed challenges to the design of high speed clock network, where clock delay and skew minimization has been a critical problem. Clock delay and skew can be optimized by good routing strategy [1][2] and effective buffer insertion [3][5]. Early research [1][2] have proposed algorithms to minimize skew by properly balance the length of the path from source to sinks. To further reduce the clock signal delay and transmission line effect, buffer insertion is necessary. The objective of buffer insertion is to find the proper buffer numbers, sizes and placement in a given clock tree.

The fix-position algorithms proposed in [6][7] insert buffers at either branching nodes in a clock tree or directly after the branching nodes. However, the optimal buffer positions for minimal delay do not necessarily reside on those places as shown by our optimal buffer position theory in this paper. Algorithms proposed in [6][8] insert the same number of buffers in each source-to-sink path. They further assume the buffers at the same level have the same size. Undoubtedly this buffer insertion strategy helps to reduce skew sensitivity to process variations [9], but this strategy requires an almost balanced routing tree topology. Otherwise, it can not effectively reduce the skew to any specified tolerance. The balanced buffer insertion scheme [10] attempts to partition the clock tree into several subtrees such that every subtree has equal path-length and all source-to-sink paths have an equal number of levels. This scheme requires an equal path length routing tree. In many real design situation, it is impossible to construct an equal-path-length tree or well-balanced tree in a routing plane if there exist pre-placed cell modules [15]. Furthermore, an equal-path-length clock tree does not guarantee the minimal delay and skew due to the fact that circuit delay depends not only on the path length but also on the circuit topology [16].

In practical IC design, the delay and skew are usually determined by the system specification. It is unnecessary to achieve zero skew [11][12]. This paper endeavors to solve the optimal buffer insertion problem from both theoretical and practical design points of view. In contrast to the previous research, we aim to establish our buffer insertion theories, methodologies and algorithms based on real circuit design settings, where the clock routing tree is neither well balanced nor equal-length.

* This research is supported in part by AirForce Office of Scientific Research grant F49620-96-1-0341, NSF NYI Award MIP-9457402 and project 69806004 supported by NSFC.

2. Definitions and Delay Model

We start by giving a synopsis of the basic definitions which will be used throughout this paper.

2.1 Definition

A tree is denoted by $T(V,E)$, where V is the set of *nodes* and E is the set of *edges*.

Definition 1 (Unbuffered routing tree): An *unbuffered routing tree* (Figure 1(a)) is a directed binary tree $T(V,E)$, which consists of wire set E and node set $V=\{\{s_0\} \cup SI \cup IN\}$, where s_0 is the unique *source* node, *IN* is the set of *branching* nodes (or *internal* nodes**)** and *SI* is the set of *sink* nodes.

Definition 2 (Buffered routing tree): After buffers have been inserted into the routing tree, the tree is called a *buffered routing tree BT* (Figure 1(b)), where the inserted buffers become the new nodes, called *buffer* nodes in the tree. The node set V of BT is then indicated by $\{\{s_0\} \cup SI \cup IN \cup BN\}$, where BN is the set of buffer nodes.

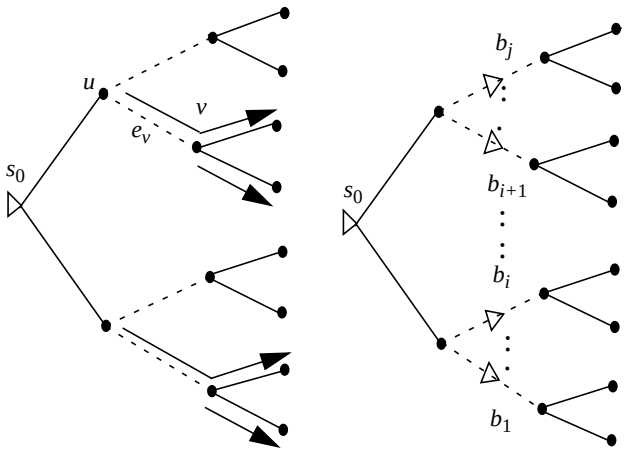
Definition 3 (Wire): A *wire* e_v ($e_v \in E$) is a directed edge (u,v) ($u,v \in V$), where the signal propagates from u to v. Node u is called the *parent* of node v, and node v is called the *child* of node u.

Definition 4 (Path): A *path* $P(v_0,v_n)$ from node v_0 to node v_n in tree T is a sequence of nodes v_0, v_1, v_2,..., $v_n \in V$ and wires e_{v_1}, e_{v_2},..., $e_{v_n} \in V$ such that e_{v_i} links v_{i-1} and v_i, where $e_{v_i} \neq e_{v_j}$, $\forall i, j, i \neq j$.

Definition 5 (Buffer layer): In a clock tree T, if k buffers are inserted into a source-to-sink path $P(s_0,s_i)$, we say that there are *k-layer* (*k-level*) of buffers on path $P(s_0,s_i)$. From source s_0 to sink s_i, the j-th (j=1,2,...,k) buffer is called *j-th layer* (*level*) buffer.

2.2 Delay Model

The delay model used in this paper is given in Figure 2. A



(a) Unbuffered routing tree (b) Buffered routing tree

Figure 1. A clock tree before and after buffer insertion.

wire in a clock tree is modeled by a distributed *RC* line (Figure 2(a)), which is further modeled by the equivalent π-model [13] as shown in Figure 2(b) (assuming properly segmented). Resistance r_e and capacitance c_e of wire e are given by Eqns. (1) and (2), respectively.

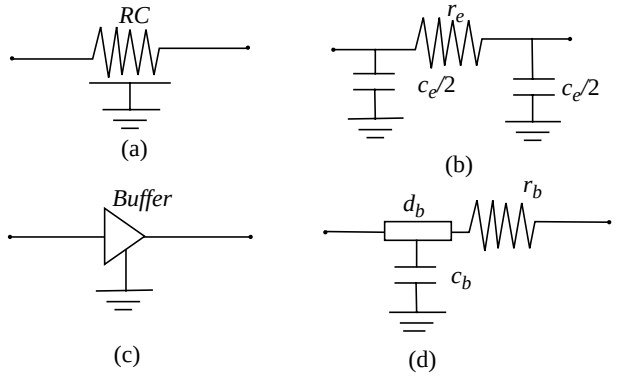


Figure 2: Delay model (a) A distributed *RC* line; (b) The equivalent π-model; (c) A clock buffer; (d) Buffer model.

$$r_e(l_e) = r_s \cdot l_e / w_e = r_0 l_e \quad (1)$$

$$c_e(l_e) = c_a l_e w_e + 2c_f(l_e + w_e) = (c_a w_e + 2c_f) l_e = c_0 l_e \quad (2)$$

where $l_e >> w_e$, and l_e and w_e are the length and width of the wire e, respectively. The constant parameters, r_s c_a and c_f, are the sheet resistance, area capacitance and fringe capacitance of a unit-width unit-length wire segment, respectively. From the above definition, terms r_0 and c_0 respectively denote the resistance and the capacitance of an unit-length wire. The buffer is modeled by a standard *RC* network (Figure 2(c,d)), where d_b, r_b and c_b are buffer's intrinsic delay, output resistance and input capacitance, respectively.

For calculating the delay of wire e_v which enters node v from its parent node, Eqn.(3) is used for the approximation, where r_{e_v} and c_{e_v} are the resistance and capacitance of the wire e_v, and $C(T_v)$ is the lumped capacitance of the subtree T_v rooted at node v.

$$\tau(e_v) = r_{e_v} \cdot \left(\frac{c_{e_v}}{2} + C(T_v) \right) \quad (3)$$

Supposing a path $P(v_0,v_n)$ consisting of a number of n wires e_{v_1}, e_{v_2},..., e_{v_n}, the path delay of $P(v_0,v_n)$ is given by Eqn. (4).

$$\tau(v_0,v_n) = \sum_{i=1}^{n} \tau(e_{v_i}) \quad (4)$$

Eqn.(4) is the well known Elmore delay [4]. Buffer delay τ_b is given by the following equation,

$$\tau_b = d_b + r_b \cdot C_b \quad (5)$$

where C_b is buffer's load capacitance, and d_b is taken as a non-zero value in our discussion and experiment.

3. Delay and Skew Minimization by Buffer Insertion

In this section, we first formulate the buffer insertion optimization problem, and then present the three-stage approach to achieve both delay and skew minimization.

Buffer Insertion Problem: Given a routed clock tree T^0 and the buffer size range from the maximum value r_{bmax} to the minimum value r_{bmin}, determine the number of the inserted buffers, their positions and sizes such that clock delay and skew are minimized.

To solve the above defined buffer insertion problem, one strategy is to fix the number of buffers first, say k buffers in each path, and then to find the optimized buffer positions and sizes. We call this problem a ***k-level buffer insertion problem***. We only need to consider this k-level buffer insertion problem since we can change the value of k from 1 to a desired number and repeat the same procedure to find the optimal value of k. Considering the number of buffers is upper bounded by a very small number in practical application, we actually do not need to run this procedure many times. Therefore, in the following, k is assumed to be a fixed value.

Figure 3 sketches our proposed three-stage buffer insertion approach for the k-level buffer insertion problem.

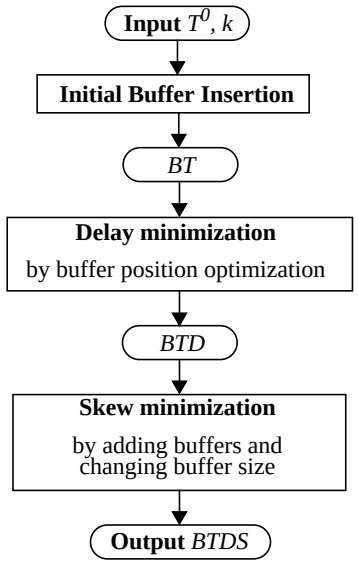


Figure 3.Three-stage clock delay and skew optimization approach.

1) Initial Buffer Insertion

The initial buffer insertion algorithm inserts k level of buffers in each source-to-sink path for a given unbuffered clock tree T^0. The algorithm generates a buffered clock tree called *BT*.

2) Delay Minimization

Without adding more buffers or changing buffer sizes, clock delay is minimized by moving buffer positions. The objective of

this stage is to generate a minimal delay buffered tree called *BTD*.

3) Skew Minimization

In this stage, the sizes and positions of the buffers on the minimum delay path are kept unchanged. We use buffer sizing, and if necessary, add more buffers to minimize skew. The result of this stage is a tree with minimal delay and skew, called *BTDS*.

In the following sections, we present details of the above three procedures.

4. Initial Buffer Insertion

The initial buffer insertion procedure was presented in our previous paper [16]. Here we briefly describe it for the purpose of a self-contained paper. The procedure is a top-down (from source to sinks) and level-by-level method to insert k level of buffers in each source-to-sink path. All the buffers are chosen the same middle size. The following theorem was proved in [16] for this algorithm.

Theorem 1: The proposed level-by-level initial buffer insertion procedure inserts the same number of buffers in each source-to-sink path.

5. Delay Minimization by Buffer Position Optimization

To achieve the minimum delay, an optimal solution of determining the number and position of buffers for driving a long uniform wire was developed in the previous work [14]. However, clock tree delay minimization by buffer placement is challenged by the fact that buffer position in one path affects the performance of the other paths. We minimize clock tree delay in a path by path fashion. To illustrate the complexity of this problem, we start by examining the problem of buffer placement in a single path in a buffered tree. We first derive a theory of the optimal buffer position and then show the need for splitting a buffer when minimizing the path delay. Based on this theory, an iterative algorithm is consequently proposed to minimize the delay of a buffered clock tree.

5.1 Single Path Delay Minimization

We now investigate how to achieve the optimal position when one buffer is moved between two branching nodes and all other buffers are fixed at their current position. Especially, we establish a theory for the optimal placement of buffers in a single path for achieving the minimal delay.

5.1.1 Buffer Position Optimization for Delay Minimization

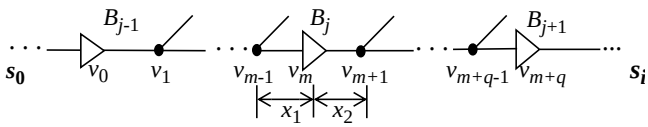


Figure 4. A buffered path.

Without loss of generality, consider one portion of a path $P(s_0,s_i)$ from source s_0 to sink s_i in a buffered clock tree, as shown in Figure 4. Between buffer B_{j-1} and B_j there are $m-1$ $(m \geq 1)$ branching nodes, v_1, v_2,..., v_{m-1}. Between buffer B_j and B_{j+1} there are $q-1$ $(q \geq 1)$ branching nodes v_{m+1}, v_{m+2},..., v_{m+q-1}. Let v_0, v_m and v_{m+q} denote the three buffer nodes. Suppose buffer B_j locates between v_{m-1} and v_{m+1}. Let x_1 and x_2 denote the wire length of $e(v_{m-1}, v_m)$ and $e(v_m, v_{m+1})$, respectively. The wire length of $e(v_{m-1}, v_{m+1})$ is a constant L_j in the given routing tree, i.e.,

$$x_1 + x_2 = L_j \tag{6}$$

Using the delay model described in Section 2, the delay $\tau(v_0, v_m)$, from buffer B_{j-1} to buffer B_j, can be written as a function of x_1 in Eqn.(7).

$$f_1(x_1) = \tau(v_0, v_m) = \alpha_1 x_1^2 + \beta_1 x_1 + \Gamma_1 \tag{7}$$

Similarly, the delay $\tau(v_m, v_{m+q})$ from buffer B_j to buffer B_{j+1} can be written as a function of x_2 in Eqn.(8).

$$f_2(x_2) = \tau(v_m, v_{m+q}) = \alpha_2 x_2^2 + \beta_2 x_2 + \Gamma_2 \tag{8}$$

Note that parameters $\alpha_1, \alpha_2, \beta_1, \beta_2, \Gamma_1, \Gamma_2$ are constants for the given path. Combining Eqns. (7) and (8) we obtain $\tau(v_0, v_{m+q})$ in Eqn. (9), the delay from v_0 to v_{m+q}, which is a quadratic function of variable x_1.

$$\begin{aligned} f(x_1) &= \tau(v_0, v_{m+q}) \\ &= \alpha_1 x_1^2 + \beta_1 x_1 + \Gamma_1 + \alpha_2 (L_j - x_1)^2 + \beta_2 (L_j - x_1) + \Gamma_2 \end{aligned} \tag{9}$$

The minimum delay of $f(x_1)$ is achieved when $f'(x_1) = 0$, i.e. the following equation holds.

$$f'_1(x_1) = f'_2(x_2) \tag{10}$$

The position of buffer B_j for minimal delay is therefore given by

$$x_1 = \frac{\beta_2 - \beta_1 + 2\alpha_2 L_j}{2(\alpha_1 + \alpha_2)} \tag{11}$$

We have the following optimal buffer position theory for delay minimization.

Theorem 2 (Optimal Buffer Position Theorem): Given a buffered clock tree, the path delay of a source-to-sink path achieves its minimum when the derivatives of all delay functions between two consecutive buffers on the path are the same.

From Theorem 2, a buffer can be continuously moved to its optimal position for delay minimization. However, if the optimal position is not in between the two adjacent branching nodes, the buffer may need to be moved over branching nodes. In such a case, the delay function is no longer a continuous function of the buffer movement. Hence, when a buffer is moved over a branching node, special treatment should be applied to ensure continuity of the delay function. In the following, we analyze this problem using the circuit theory.

5.1.2 Buffer Splitting Theory

Consider a generic case shown in Figure 5, according to Theorem 2 buffer B_j will be moved continuously to the right along the wire in order to reduce the path delay. But when the buffer hit the branching node $v_{m'}$, further movement needs to be treated specifically. In this subsection, we discuss how to split a buffer into two smaller buffers, when it is moved down through a branching node to maintain the continuity of the delay function. In figure 5, before buffer splitting, for $\tau(v_0, v_{m+q})$ the delay portion contributed by buffer B_j's output resistance r_{b_j} and input capacitance c_{b_j} is given by

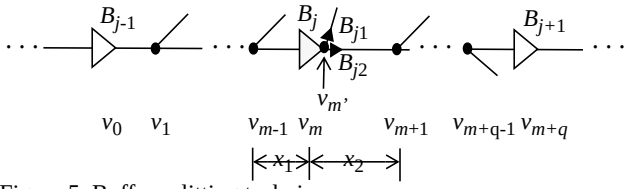


Figure 5. Buffer splitting technique.

$$\tau_1 = R_p c_{b_j} + r_{b_j}(C_{p1} + C_{p2}) \tag{12}$$

where R_P is the resistance of path $P(v_0,v_m)$, C_{P1} and C_{P2} are

lumped capacitance of the two subtrees rooted at node $v_{m'}$. After buffer B_j is moved onto branches 1 and 2 directly after the branching node $v_{m'}$, it is split into two buffers B_{j1} and B_{j2}. For $\tau(v_0, v_{m+q})$ the delay portion τ_2 contributed by buffer B_{j2}'s input capacitance $c_{b_{j2}}$ and output resistance $r_{b_{j2}}$ is

$$\tau_2 = R_p(c_{b_{j1}} + c_{b_{j2}}) + r_{b_{j2}} C_{P2} \quad (13)$$

where $c_{b_{j1}}$ is the input capacitance of buffer B_{j1}. To ensure the continuity, τ_2 should equal to τ_1, which results in the following two conditions in Eqn.(14) and Eqn.(15).

$$c_{b_j} = c_{b_{j1}} + c_{b_{j2}} \quad (14)$$

$$r_{b_j}(C_{P1} + C_{P2}) = r_{b_{j2}} C_{P2} \quad (15)$$

Suppose $C_{P1} = \gamma C_{P2}$. We obtain the size relationship of the two split buffers in Eqn. (16).

$$W_{b_{j1}} = \frac{\gamma}{1+\gamma} W_{b_j} \qquad W_{b_{j2}} = \frac{1}{1+\gamma} W_{b_j}, \; L_{b_{j1}} = L_{b_{j2}} = L_{b_j} \quad (16)$$

where L_{b_j}, $L_{b_{j1}}$, $L_{b_{j2}}$ denote the transistor gate length of buffer B_j, B_{j1} and B_{j2}, and W_{b_j}, $W_{b_{j1}}$, $W_{b_{j2}}$ denote their gate width.

Theorem 3 (Buffer Splitting Theorem): In a buffered clock tree, when a buffer is moved down over a branching node, if it is split into two buffers with their sizes satisfying Eqn. (16), all path delays in the clock tree remain the same.

5.1.3 Algorithm for Single Path Delay Minimization

Using Theorem 2 and 3, we develop an iterative algorithm for single path delay minimization. The algorithm repeatedly selects a buffer on the path from source to sink and tests if the path delay can be reduced by changing buffer position. If the delay is reduced, the buffer is moved to the new position. Otherwise, it stays in its original position. When the last buffer on the path is tested, the algorithm goes back to the first buffer and starts a new iteration until the delay can not be further reduced. Formally, the single path delay minimization (*SPDM*) algorithm is given in Table 1.

5.2 Delay Minimization for Clock Tree

The algorithm for tree delay minimization is based on the single path delay minimization. However, tree delay minimization is much more difficult to handle than single path delay minimization, because one path minimization affects the performance of the others which may have already been minimized. An effective algorithm should be developed to ensure the convergence of the minimization process.

5.2.1 Tree Delay Minimization Algorithm

Suppose there are q paths in a buffered clock tree. Let P_{min} denote the minimal delay path and τ_{min} denote the minimal delay. Let P_i denote the i-th path to be minimized and τ_{P_i} be the delay of path P_i. The algorithm is as follows.

Step1. Sort all paths in *BT* (buffered tree) according to their delay in an ascending order $\tau_{P_1}(\tau_{min}) < \ldots < \tau_{P_i} < \ldots < \tau_{P_q}$;

Step2. For $i = 1$ to q

Minimize delay of P_i using the single path delay minimization algorithm with buffer moving constraints (discussed in Section 5.2.2);

Step3. Repeat Step1 and Step2 until τ_{min} can not be further reduced.

Table 1. Algorithm of single path delay minimization.

Input: Path $P(s_0,s_i)$ (There's k buffers $B_1, B_2,..., B_k$ on this path from source s_0 to sink s_i);

ΔX : buffer moving step;

Output: Optimized path $P(s_0,s_i)$;

```
Procedure PathDelayMinimization (P(s0,si), k)
    τmin ← Calculate path delay;
    move_flag ← 1;
    while (move_flag = 1)
        move_flag ← 0;
        for i ← 1 to k do
            τ ← Calculate path delay, if buffer Bi is moved up ΔX;
            if τ < τmin then
                Move buffer Bi up ΔX;
                move_flag ← 1;
            else
                τ ← Calculate path delay, if buffer Bi is moved down ΔX;
                if τ < τmin then
                    Move buffer Bi up ΔX;
                    move_flag ← 1;
                end if;
            end if;
        end for;
    end while;
end Procedure;
```

5.2.2 Moving Constraints for Monotonous Delay Reduction

Suppose P_i is the current path for delay minimization. To ensure the delay monotonously decreasing and the convergence of the algorithm, three constraints are enforced in the buffer moving process:

A. If a buffer on path P_i is also on path P_{min}, it can not be moved.

B. From source to sink on path P_i, the first buffer which is not on path P_{min} can not be moved down along the path.

C. On path P_i other buffers which are not in case A and B, can be moved both up and down, but can not be moved up through the branching nodes and will be split into two buffers when moved down through the branching nodes.

Figure 6 illustrates the above constraints. In this example, $P(s_0, s_1)$ is the minimal delay path P_{min}. Suppose we optimize path $P(s_0, s_i)$ by moving buffers on this path. According to the buffer moving constraints, buffer B_{11} can not be moved. Buffer B_{i1} can only be moved towards branching node v_1, but can not be moved through v_1. Buffer B_{i1} can not be moved towards branching node v_4, since such a movement will increase the capacitive load of the subtree rooted at buffer B_{11}, which will increase the previously optimized minimum delay. Buffer B_{i2} can be moved between v_4 and v_5, but can not be moved over v_4, and will be split into two buffers when moved through v_5.

The tree delay minimization algorithm optimizes the tree delay in a path by path fashion using the single path delay minimization algorithm (*SPDM*). After the tree delay minimization we obtain a buffered tree with minimal delay, called *BTD*.

6. Skew Minimization Strategy

Skew minimization is achieved by increasing buffer size or adding more buffers (if necessary) to the paths such that delays on these paths decrease towards the minimum delay. To ensure this

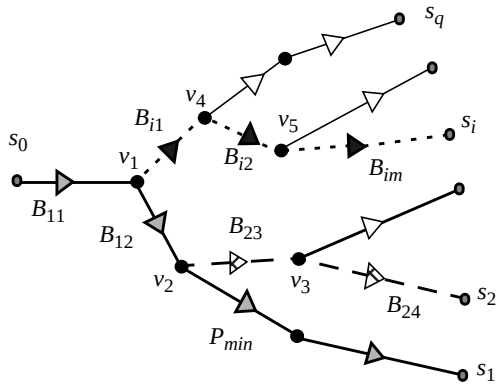


Figure 6: Delay minimization by buffer position optimization.

strategy do not increase the minimum delay, the sizes and positions of the buffers on the minimal delay path in *BTD* are kept unchanged. Also we will setup several constraints for buffer sizing and buffer adding to guarantee the convergence of the algorithm.

6.1 Skew Minimization Algorithm

Step 1. Sort all the paths in *BTD* according to their delay in an ascending order $\tau_{P_1}(\tau_{min}) < \ldots < \tau_{P_i} < \ldots < \tau_{P_q}$;

Step 2. For i = 2 to q

On the current path P_i, find the subpath for buffer sizing or buffer adding. Reduce the delay of path P_i with respect to the minimum delay τ_{min} by sizing or adding buffers;

Step 3. Repeat Step1 and Step 2 until the skew specification is met.

The above algorithm minimizes skew path by path. Once a path is optimized, all the buffers on that path will have their sizes and positions fixed such that later optimized paths will not increase the delay of the previously optimized paths.

For example, in Figure 7, $P(s_0, s_1)$ is the minimal delay path. Buffer sizing has been done for path $P(s_0, s_q)$. Suppose $P(s_0, s_i)$ is the current path under consideration of sizing. Since buffer B_{11} is on the minimal delay path and the size of buffer B_{i1} has been optimized, on subpath $P(v_i,s_i)$ only buffers B_{ij} and B_{im} are the candidate buffers, which could be done sizing for the skew optimization.

Eqn. (17) tests whether the delay of the considered path can be reduced to satisfy the specified skew tolerance by setting all the candidate buffers to the maximum buffer size r_{bmin}. If Eqn. (17) holds, changing buffer size is enough to meet the skew specification. Otherwise, more buffers need to be added into this path.

$$\sum_{j=1}^{m} (r_{bj} - r_{bmin}) C_j \geq \tau_{p_i} - \tau_{min} - \tau_{skew} \qquad (17)$$

In Eqn.(17), suppose there are m candidate buffers for buffer sizing. Term C_j denotes the lumped capacitance of the subtree rooted at the j-th buffer, r_{bj} denotes the buffer's output resistance, and τ_{skew} is the specific skew tolerance. If buffer sizing suffices, specific buffer sizing algorithm is given below.

6.2 Buffer Sizing Procedure

Suppose there are m candidate buffers on the subpath for sizing. Along this subpath, from top to bottom, the procedure increases the candidate buffer size one by one to the maximum until the current path delay is smaller than the minimal delay of the tree. Then, the size of this current buffer is determined by Eqn. (18),

$$\sum_{j=1}^{n-1} (r_{bj} - r_{bmin}) C_j + (r_{bn} - \tilde{r}_{bn}) C_n = \tau_{P_i} - \tau_{min} \qquad (18)$$

where $n-1$ $(1 \leq n \leq m)$ is the number of buffers having been increased to the maximum size r_{bmin} and $\tilde{r}_{bn}$ is the size of the current buffer under consideration (labeled as the *n-th* buffer). Note that due to the buffer structure is the cascaded inverters [17], when buffer size is changed, its input capacitance remains the same, only its intrinsic delay and output resistance change. And we assume buffer intrinsic delay varies directly as its output resistance.

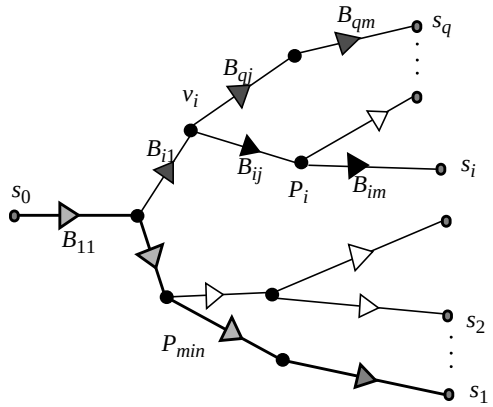


Figure 7: Buffer sizing for skew minimization.

6.3 Adding Buffer Strategy

When all buffers in the candidate subpath are increased to their maximum size and the skew is still beyond the specification, additional buffers are added to the subpath. A four-step strategy is proposed for the buffer adding algorithm.

Step 1. Apply the Layer Defined Initial Buffer Insertion algorithm to add one more layer buffer to the candidate subpath;

Step 2. Apply the tree delay minimization algorithm to optimize the positions of the buffers on the candidate subpath;

Step 3. Apply the buffer sizing algorithm to optimize the sizes of the buffers on the subpath;

Step 4. Repeat Step 1 to Step 3 until the tolerable skew is within the specification.

7. Results and Conclusions

The presented algorithm was implemented in C language on SPARC 20 workstation. The clock routing is generated using the algorithm in [15], where routing obstacles exist, besides the routing trees are not well balanced and not equal length. Table 2 shows the technology parameters used in our experiment. We tested a 6mm x 6mm chip and a 1cm x 1cm chip. In each test case, the routing trees have the number of sinks 50, 100, 300 and 500, respectively.

Table 3 presents results of delay and skew before and after buffer insertion. Different number of the inserted buffer layers were tested and the optimal buffer layers were found. In the table, the delay is measured as the signal propagation time from the source to the sinks, i.e., the summation of the Elmore delays of the subtrees and the buffer delays along a source-to-sink path. The results show that the reduction of both delay and skew using the proposed method are in one order of magnitude. For all of the tested cases the skew is reduced to within the scope of 100ps which is necessary for running the clock network at multi-GHz.

In this paper, we derived an optimal buffer placement theory for delay minimization based on the Elmore delay model. We

Table 2: Technology parameters.

Sheet Resistance 0.14 Ω/❑	Area Capacitance 0.08 fF/μm^2	MaxBuffer output R 5 Ω	MaxBuffer input C 0.01 pF	MaxBuffer intrinsic delay 100 ps
Wire Width 10 μm	Fringe Capacitance 0.03 fF/μm	MinBuffer output R 100 Ω	MinBuffer input C 0.01 pF	MinBuffer intrinsic delay 30 ps

Table 3: Results before and after buffer insertion.

Chip	Sink#	Longest Path Length (cm)	InitiallyInserted Buffer Layer	Before Buffer Insertion Delay (τ_0)	Skew (τ_{s0})	After Delay&Skew Minimization Delay (τ)	Skew (τ_s)	Delay&Skew Improvement Impv1	Impv2
Chip1	50	2.24	4	4422.08	430.62	707.48	47.25	6.25	9.11
	100	2.06	4	5668.06	169.39	674.31	52.27	8.41	3.24
	300	2.01	5	11445.65	124.60	710.10	48.82	16.12	2.55
	500	2.32	4	14794.05	1248.53	660.70	72.32	22.39	17.26
Chip2	50	3.39	4	9932.96	917.30	790.55	58.70	12.56	16.63
	100	3.42	5	14727.49	891.29	831.38	66.97	17.71	12.41
	300	3.39	4	27055.19	659.11	1106.18	87.89	24.46	7.50
	500	3.27	4	36485.87	889.18	1325.29	84.11	27.53	10.57

*all delay and skew data are in ps. $Impv1 = \tau_0/\tau$, $Impv2 = \tau_{s_0}/\tau_s$

show that the optimal buffer placement for delay minimization is achieved when all delay functions have the same derivative values. Consequently, we developed a delay minimization algorithm which gives the minimal clock delay. A skew minimization scheme to minimize the skew of the clock signal is further expatiated. To achieve the minimal skew, we also derived a formula to calculate the optimal size of the inserted buffers in the meantime. Since our approach is carried out in the manner of path by path, it can be easily used for a more general case where the signal arriving time to each individual sink is specified to be different.

Acknowledgment

The authors wish to thank Mr. Haksu Kim for his valuable discussions and providing us with the test input examples.

References

[1] M. A. B. Jackson, A. Srinivasan and E. S. Kuh, “Clock routing for high-performance ICs”, *27th ACM IEEE Design Automation Conference*, pp.573-579, 1990.
[2] A. Kahng, J. Cong and G. Robins, “High-performance clock routing based on recursive geometric matching”, *28th ACM IEEE Design Automation Conference*, pp.332-337, 1991.
[3] B. Wu and N. Sherwani. “Effective Buffer insertion of Clock Tree for High-Speed VLSI Circuits”. *Microelectronics Journal*, 23:291-300, July 1992.
[4] W.C. Elmore, “The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifier,” *J. Applied Physics*, 19, pp. 55-63, 1948.
[5] J. Lillis, C.K. Chen and T.T. Lin, “Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model”, *Proc. IEEE Int. Conf. on Computer-Aided Design*, pp. 138-143,1995.
[6] S. Pullela, N. Menezes, J. Omar and L. Pillage, “Skew and Delay Optimization for Reliable Buffered Clock Trees”, *Proc. IEEE Int. Conf. on Computer-Aided Design*, pp. 556-562,1993.
[7] L.P.P.P. van Ginneken. “Buffer placement in distributed RC-tree networks for minimal Elmore delay”. *International Symposium on Circuits and Systems*, pp. 865-868, 1990.
[8] Y. P. Chen and D.F. Wong, “An algorithm for zero-skew clock tree routing with buffer insertion”, *Proc. European on Computer-Aided Design*, pp.219-223, 1994.
[9] J. Cong, L. He, C.K. Koh and P. H. Madden. “Performance optimization of VLSI interconnect layout”. *INTEGRATION, the VLSI Journal*, 21:1-94, 1996.
[10] Joe G. Xi and Wayne W. M. Dai, “Buffer Insertion and Sizing Under Process Variations for Low Power Clock Distribution”, *ACM/IEEE Design Automation Conference*, pp. 491-496, 1995
[11] F. Anceau, “A synchronous approach for clocking VLSI systems”, *IEEE J. Solid-State Circuits*, vol. SC-17, pp. 51-56, 1982.
[12] J. P. Fishburn, “Clock skew optimization”, *IEEE Trans. Computer*, vol. 39, pp. 945-951, July 1990.
[13] N. Sherwani, “Algorithms for VLSI Physical Design Automation,” 2nd edition, *Kluwer Academic Publisher*s, pp. 400-402, 1998.
[14] S. Dhar and M.A. Franklin, “Optimum Buffer Circuits for Driving Long Uniform Lines”, *IEEE J. of Solid-State Circuits*, vol. 26(no.1), pp.32-40, Jan. 1991.
[15] H. Kim and D. Zhou, “An Automatic Clock Tree Design System for High-Speed VLSI Designs: planar clock routing with the treatment of obstacles”, *International Symposium on Circuits and Systems*, 1999 to appear.
[16] D. Zhou, W. Li and X. Zeng, “An Effective Buffer Insertion Algorithm for High-Speed Clock Network”, submitted to *IEEE Design Automation Conference*, 1999.
[17] D. Zhou and X.Y. Liu, “On the Optimal Drivers of High-Speed Low Power ICs”, *International Journal of High Speed Electronics and Systems*, vol. 7(no.2), pp. 287-303, 1996.