Algorithms for Non-Hanan-based Optimization for VLSI Interconnect under a Higher Order AWE Model

Jiang Hu and Sachin S. Sapatnekar

jhu@mountains.ee.umn.edu and sachin@mountains.ee.umn.edu

Department of Electrical and Computer Engineering

University of Minnesota

Minneapolis, MN 55455, USA

Tel: 612-625-0025, Fax: 612-625-4583

Abstract

To improve the performance of critical nets where both timing and wire resources are stringent, we integrate buffer insertion and driver sizing separately with non-Hanan optimization and propose two algorithms: BINO (simultaneous Buffer Insertion and Non-Hanan Optimization) and FAR-DS (Full-plane AWE Routing with Driver Sizing). For BINO, we consider the realistic situation that buffer locations are restricted to a limited set of available spaces after cell placement. The objective of BINO is to minimize a weighted sum of wire and buffer costs subject to timing constraints. To achieve this objective, we suggest a greedy algorithm that considers two operations independently: iterative buffer insertion and iterative buffer deletion. Both are conducted simultaneously with non-Hanan optimization until the improvement is exhausted. For FAR-DS, we investigate the curvature property of the sink delay as a function of both connection location and driver stage ratio in a two-dimensional space. The objective of FAR-DS is to minimize a weighted sum of wire and driver cost while ensuring that the timing constraints are satisfied. Based on the curvature property, we search for the optimal solution in the continuous two-dimensional space. In both BINO and FAR-DS, a fourth order AWE delay model is employed to assure the quality of optimization. Experiments of BINO and FAR-DS on both IC and MCM technologies showed significant cost reductions compared with SERT and MVERT in addition to making the interconnect to satisfy timing constraints.

1 Introduction

As the VLSI technology develops into the deep sub-micron era, the interconnect resistance is no longer negligible and its performance plays a critical role to the whole circuit. As the result, many efforts [1–18] have been carried out in recent years to improve the interconnect performance. According to the classification in [1], these works have evolved along three major aspects: the *delay model*, the *objective formulation* and the *solution space*. The progress on each of these aspects will be briefly reviewed as follows.

When the interconnect resistance was not significant, it could be simply modeled as a lumped capacitance that is proportional to the wire length. Therefore, in early research, the interconnect performance criterion was purely geometric and focused on wire length based objectives such as reducing the routing radius and the total wire length. As wires have become longer and thinner, this geometric evaluation no longer suffices to reduce interconnect delay as resistive effects become significant. A more elaborate *delay model* is necessary to augment wire length considerations in performance evaluation. The Elmore delay [19] model has been widely used due to its simplicity and high fidelity [2]. Its simplicity not only removes the need for large amount of computation, but also provides a platform on which many theoretical properties can be derived and exploited. One major Elmore delay based routing method, SERT [2], grows the routing tree in a greedy fashion to minimize the source-sink delay. Another Elmore delay application, the P-tree algorithm [3], first searches for a good permutation of the sinks and then limits the solution space to the topologies induced by this permutation. In later work, the drawbacks of Elmore model have been addressed and second [5] and third order [6] models have been applied. Most recently, the work in [1] suggested a table lookup method to remedy the deficiencies of the Elmore delay model.

With regard to the *objective formulation*, the total wire length (area) and delay are usually the major targets. Minimizing total wire length can reduce the cost, power consumption and improve the routability. All of these advantages lead to the use of area minimization as a common baseline for objective formulations. For delay reduction, there are many forms in which the objective may be stated, including minimizing either a weighted sum of sink delays, or the maximum delay, or the critical sink delay. As a more appropriate formulation, the research in [5, 9, 10] focuses on satisfying the timing specification in an effort to trade off the unnecessary delay reduction into area minimization.

The *solution space* of nodes in the routing tree has long been restricted to the Hanan grid since it simplifies the problem nature, and it can be proven that optimal solutions lie only on Hanan grid points if the unconstrained objective is to minimize the wire length or a weighted sum of sink delay. However, if we formulate the objective so as to satisfy the timing constraints, the optimal Steiner points are very likely to lie at the non-Hanan grid points, as indicated in [10]. The work of [10] developed the MVERT algorithm, which exploits the piecewise concavity of delay violation functions to search for the optimal Steiner points. Its experimental results showed that expanding the solution space to non-Hanan points can significantly reduce the wire cost.

In this paper, we continue the effort of non-Hanan optimization to deal with the condition where both timing and wire resources are stringent. We integrate buffer insertion and driver sizing with non-Hanan optimization in separate formulations to further improve the interconnect performance. These two approaches resemble each other in term of their algorithmic skeleton, although the nature of the problems is different.

Buffer insertion is a promising technique [12–17] that is essential for large nets. Most of the methods in [12–17] are implemented through dynamic programming in a bottom-up fashion. However, all of these methods have been restricted to only Hanan grid routing. Moreover, each of these approaches neglects the effects of restrictions on the buffer locations, i.e., it is assumed that buffers can be inserted in any arbitrary position as long as they can improve the interconnect performance. In real situations, this is not always permissible because the optimal buffer location may already have been occupied by other cells and it is undesirable to disturb the placement. Most recently, the work of [18] takes the restrictions to buffer locations into consideration and suggests an exact algorithm for two-pin nets. The problem environment we consider here is a limited set of buffer spaces where buffers are to be inserted into the interconnect after the placement stage. The concept of soft edge is employed to increase the possibility that a buffer space is exploited. We guide each move in the optimization in a greedy fashion and conduct buffer insertion and non-Hanan optimization (BINO) simultaneously and iteratively until no further improvements are possible.

Another effort in our work is simultaneous driver sizing and non-Hanan optimization (FAR-DS). We have investigated the curvature properties of the delay as a function of the connection location and driver stage ratio in a two-dimensional space under the Elmore delay model. Though the Elmore

model may be poor for specific points, it still provides a valid prediction of qualitative properties [2]. According to the solution region properties, we suggest two search schemes to find the optimal solution in the objective that can minimize a weighted sum of the wire cost and the driver cost, while satisfying the timing constraints. In both FAR-DS and BINO, we use a fourth order AWE delay model [21] to assure the integrity of the optimization.

This paper is organized as follows. In section 2, we will introduce the concept of a soft edge, review some background on non-Hanan optimization and the motivation for using a fourth order AWE model. Section 3 will describe the problem environment and formulation of BINO (simultaneous Buffer Insertion and Non-Hanan Optimization). The problem properties and formulation of FAR-DS (Full-plane AWE Routing with Driver Sizing) will be investigated in section 4. Section 5 will present the algorithms for BINO and FAR-DS, followed by a complexity analysis in section 6. Finally, the experimental results will be discussed in section 7.

A list of notational terms used in this work is listed below:

- q_i: required arrival time for sink v_i.
- t_i: the calculated delay for sink v_i in the routing tree.
- u_i: delay violation of sink v_i, given by $u_i = t_i - q_i$.
- W: total wire length for a routing tree.
- C_g: gate capacitance of minimum sized driver.
- C_d: drain capacitance of minimum sized driver.
- P: a set of given buffer spaces.
- m: number of buffer spaces.
- B: the set of buffers inserted in the tree.
- r: resistance per unit length for interconnect.
- c: capacitance per unit length for interconnect.

- R_0: resistance of minimum sized driver.
- ρ: stage ratio for cascaded drivers.
- h: number of stage for cascaded drivers.
- n: number of sinks.
- γ: weighting factor for total wire capacitance.
- CC: the closest connection point between a node and an edge.
- l_{ij}: Manhattan distance between two nodes v_i and v_j.
- T_k: a subtree rooted at node v_k.

2 Preliminaries

2.1 Soft edges

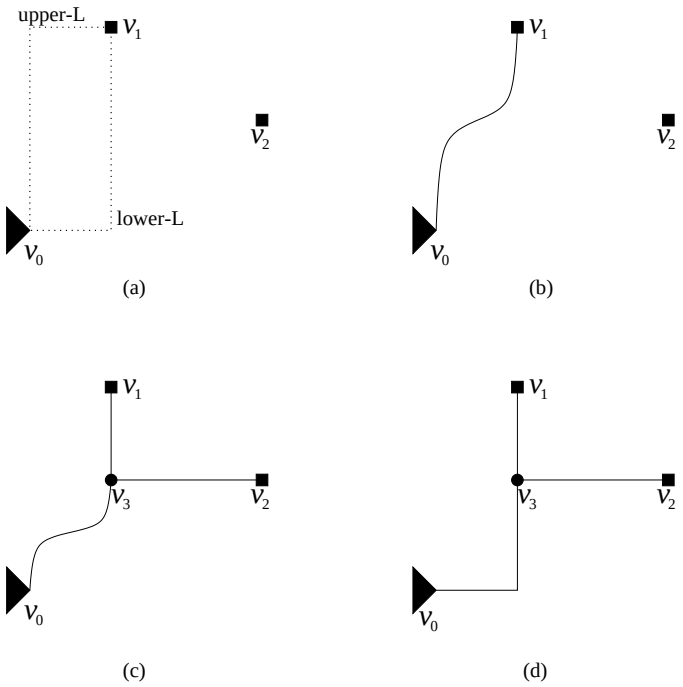

Figure 1: Routing with soft edges.

A routing tree T is described by a set of nodes $V = \{v_0, v_1, v_2...\}$ and a set of edges $E = \{e_1, e_2...\}$. Generally, we refer to v_0 as the source. The location for a node v_i is specified by its coordinates x_i and

y_i, and an edge in E is uniquely identified by the node pair (v_i, v_j). The edge length is given by the Manhattan distance between the two nodes, which is $|x_i - x_j| + |y_i - y_j|$.

Since a routing tree is built in rectilinear space, each edge must be either horizontal or vertical. We introduce another type of edge, a *soft edge*, whose orientation is not specified until the tree construction and optimization are completed. The concept of soft edge will be explained through a simple example in Figure 1. In this example, a source v_0 and two sinks v_1 and v_2 are given, and a minimum Steiner tree is to be constructed on this node set by adding one node to the tree at a time. We may begin by connecting v_1 to v_0. In rectilinear space, there are two L-shaped connection options, shown by the dashed lines in Figure 1 (a); one bend is required for each connection. These two options will provide different results when v_2 is connected to the tree, and a lower-L connection for v_1 is obviously better in this example, since it provides a shorter wire length. However, when the number of sinks is large, it may be very hard to see which option is better. As suggested in [2], the decision can be deferred until all the sinks are joined into the tree, and we can connect v_0 and v_1 by a soft edge that is formally defined as follows.

Definition 1: A *soft edge* is an edge connecting two nodes $v_i, v_j \in V$, such that:

1. $x_i \neq x_j$ and $y_i \neq y_j$,
2. its edge length $l_{ij} = |x_i - x_j| + |y_i - y_j|$,
3. the edge route between v_i and v_j is not determined.

The soft edge connection between v_0 and v_1 is shown in Figure 1(b). We will refer to the traditional edges in rectilinear tree with fixed orientations as *solid edges*. The sink v_2 is connected to the routing tree at the closest connection (CC) point, defined below, between v_2 and edge (v_0, v_1).

Definition 2: The *closest connection* (CC) *point* between a node v_k and an edge (v_i, v_j) is defined by its coordinates x_{CC} and y_{CC} such that: $x_{CC} = median(x_i, x_j, x_k)$ and $y_{CC} = median(y_i, y_j, y_k)$.

Note that in definition 2, the edge (v_i, v_j) can be either a soft edge or a solid edge.

If the CC point does not coincide with either of v_i, v_j and v_k, a Steiner node is introduced at the CC point. In the example of Figure 1, Steiner node v_3 is introduced. After this connection has been made, edge (v_3, v_1) and (v_3, v_2) are solid edges. Since all of the sinks have been joined to the routing tree, we can convert the soft edge (v_0, v_3) in Figure 1 (c) into an L-shaped connection, as shown in Figure 1 (d).

The advantage of using soft edges is that it provides a set of flexible connection choices for subsequent

routing steps and avoids premature decisions. In section 3, we will show that the use of soft edges also has some advantages that aid our buffer insertion algorithm.

2.2 Background on non-Hanan optimization

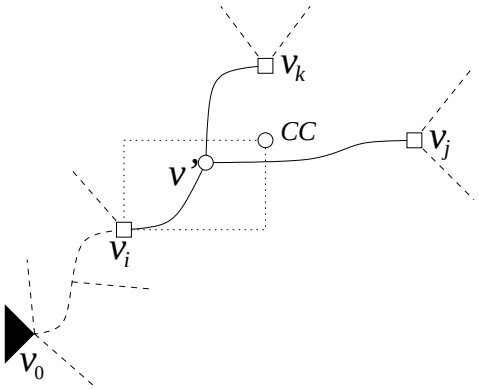

Figure 2: A general situation where node v_k is to be connected to an edge (v_i, v_j).

The concavity property exploited in non-Hanan optimization [10] depends on the concept of maximal segment [2], which requires the assignment of a definite orientation (horizontal or vertical) for each edge. Although by definition, the orientation for a soft edge is not fixed, the concavity property continues to hold for a soft edge, and we can extend the philosophy of non-Hanan optimization to general edges including both solid and soft edges.

For a general form of a routing tree, shown in Figure 2, let us consider the process of obtaining an optimal connection between node v_k and edge (v_i, v_j). Note that v_i, v_j and v_k can either be a sink, or a Steiner node with a degree of at least three. The dashed lines are other nodes and edges of this routing tree, and CC represents the closest connection point between v_k and (v_i, v_j). It can be easily seen that any connection that is downstream of CC cannot give an optimal solution [2]. More specifically, we wish to search for an optimal connection point within the bounding box defined by v_i and CC.

Suppose we connect v_k to (v_i, v_j) at point $v'(x', y')$. Let z be the Manhattan distance from v' to v_i, i.e.,

$$z = |x' - x_i| + |y' - y_i| \tag{1}$$

For convenience, we overload CC as its Manhattan distance to v_i. We can obtain the following conclusion, whose derivation is given in the appendix:

Theorem 1: Under the Elmore delay model, the delay violation at any sink in the routing tree is a concave function with respect to z.

Corollary 1: Under the Elmore delay model, for any interval $[z_l, z_r] \subseteq [0, CC]$, if the delay violation at a sink is positive when the connection point is either at z_l or z_r, then the delay violation at this sink is positive when the connect point is at any location in this interval.

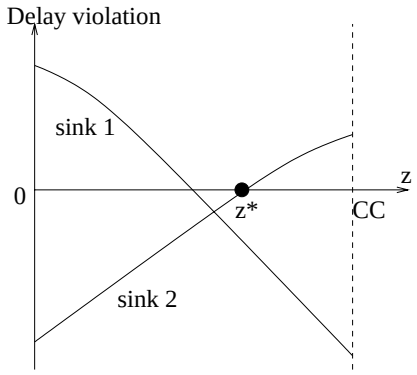

Figure 3: Delay violation function vs. Manhattan distance z of connection point.

In Figure 3, the delay violation functions of a two-sink net are depicted. If the objective is to minimize wire cost subject to timing constraints, the optimal connection (Steiner) point here is a point with a non-positive maximum delay violation, lying as close to CC as possible; for this particular example, this corresponds to z^*. As in this example, the optimal connection point is, in general, likely to be a non-Hanan point.

The work of [10] showed this advantage of using non-Hanan points and proposed the MVERT (Maximum delay Violation Elmore Routing Tree) algorithm to perform the non-Hanan optimization globally for an interconnect routing tree. Based on properties similar to *Corollary 1*, MVERT finds the optimal connection point through a quasi-binary-search and obtains significant wire cost reductions. In fact, non-Hanan optimization can also help interconnect to meet timing constraints besides affecting wire cost reductions. In the example of Figure 3, it is probable that only non-Hanan points can satisfy the timing constraints for both sinks.

Non-Hanan optimization on soft edges requires additional specifications, since the search on delay violation functions only provides an optimal Manhattan distance z and this does not, in general yield a unique point. For simplicity, we choose x' and y' both to be proportional to z. For example, if

$z = 0.3CC$, we will choose $x' = 0.3x_{CC}$ and $y' = 0.3y_{CC}$.

2.3 The motivation for using fourth order AWE

As interconnect wires become increasingly thinner and longer, the interconnect resistance may overshadow the driver resistance. Consequently, the downstream capacitance is shielded to the driver resistance by the interconnect resistance. This effect is called resistive shielding [23]. The Elmore delay does not correctly take the resistive shielding effect into account and tends to overestimate the delay. This error can be remarkably large, especially for the stub situation (i.e., when a sink that is close to the source co-exists with a much longer wire), where the Elmore delay can be several times larger than the actual delay.

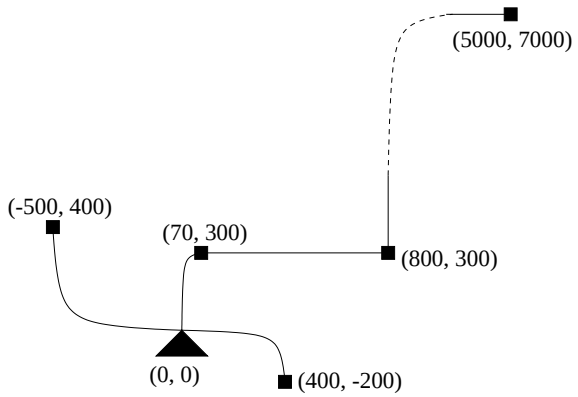

Figure 4: A routing tree on which Elmore delay gives large errors.

Table 1: A comparison of the Elmore and the 4th order AWE delays with SPICE

Dist.	SPICE	Elmore	Error	4th AWE	Error
370	13.6	52.5	286%	12.8	-6%
600	9.5	39.8	319%	8.9	-6%
900	10.7	40.5	279%	10.5	-2%
1100	26.2	77.4	195%	25.5	-3%
12000	283.2	257.5	-9%	282.4	-0.3%

Table 1 shows an example of a net with five sinks to illustrate the inaccuracy of the Elmore delay. The routing topology of this net is illustrated in Figure 4. The load capacitance is the same for each sink. The delays at all sinks are computed using the Elmore formula, fourth order AWE and a SPICE transmission line model, and the percentage errors relative to SPICE are calculated. The Manhattan distance from each sink to the source are also listed for reference. We can see that the error of Elmore

delay can be over 300% and the delay from fourth AWE is clearly superior. In fact, as the minimum feature size shrinks, this trend will become more and more severe.

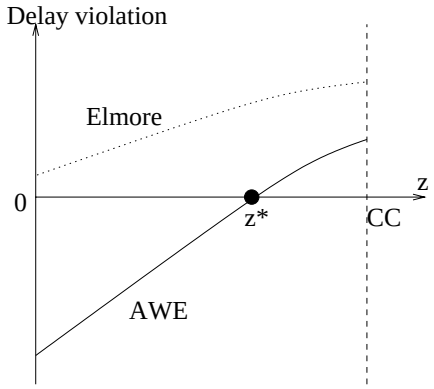

Figure 5: An example where using the Elmore delay and a higher order AWE delay may result in a different connection choice.

To see how this will affect non-Hanan routing, consider the graph in Figure 5. The graph plots the delay violation function against the location of the connection point, z, as pictured in Figure 3. The dotted curve indicates the Elmore delay while the solid curve represents the fourth order AWE result. The solution corresponds to the point closest to CC where the delay violation function is negative or zero. For the Elmore delay, which overestimates the delay near the source, no solution is found, whereas an actual solution exists and corresponds to z^*.

On the other hand, we have observed that the Elmore model tends to under-estimate delay at sinks far from the source[1]. This may lead to the opposite error, as can be seen in the last row of Table 1. This under-estimation may result in over-reduction of cost while the timing constraints have not been satisfied yet. On the whole, a higher order model is greatly superior to the Elmore model in handling non-Hanan points.

The reason that we choose fourth order instead of a second or third order model is that a second order yields less accuracy and for many examples that we tried, and we found that the third order model induces positive poles more often. The additional computation cost of fourth order AWE as compared to a second order model is minor.

In the computation of fourth order AWE delay, we first use the RICE algorithm [22] to obtain the

[1]The Elmore delay is theoretically proven to be an upper bound on the delay of an RC network in [24]. However, in practice, greater accuracies are obtainable by multiplying the Elmore delay formula of [20] by a factor of $\ln 2$, and we refer this quantity as the "Elmore delay" in our discussion, and this may be either optimistic or pessimistic.

moments. We solve the denominator of Padé approximation result, which is a fourth order polynomial, using a closed-form formula to obtain the poles. After an inverse Laplace transformation, the time-domain exponential functions are expanded about the Elmore delay to fourth order Taylor series polynomials. A closed-form solution to a fourth order polynomial exists and may be used to calculate the delay value. Since the Elmore delay may be far off from correct value, sometimes the expansion about Elmore delay may still cause significant error, though it is much smaller than the error from the Elmore delay. We restrict such error by another iteration with expansion about the result from the first iteration. This process is iterated until convergence, and we found that we always converged within three iterations. This method is related to the Newton-Raphson root-finding method: the Newton-Raphson method uses a first order Taylor series in each iteration, and our method uses a fourth order expansion instead.

3 The problem environment and problem formulation for BINO

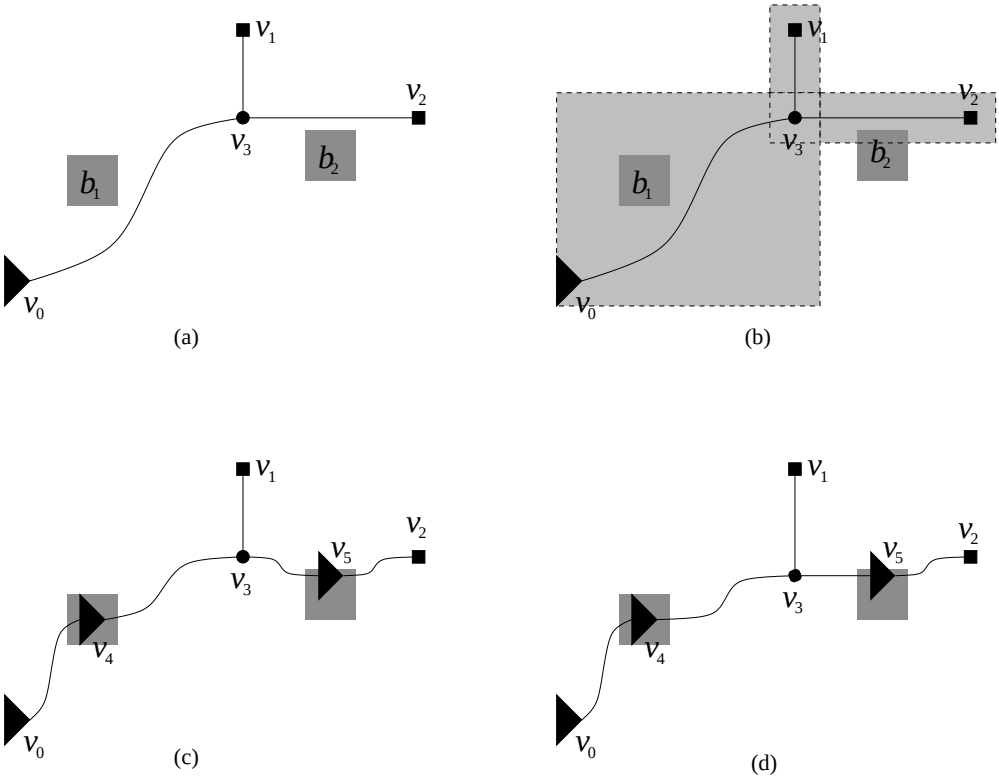

Figure 6: Buffer spaces, the territory box and their applications in buffer insertion.

The BINO algorithm is applied in a post-placement scenario where buffer insertion is possible, but it is preferable to do so in regions that are left unoccupied by any cells, so as not to disturb the placement. For MCM technology, a buffer location is desired to be within a chip and close to its chip bond pads, because it is not cost effective to insert a buffer either on the substrate between chips or within a chip but far from any bond pad. The input to BINO then includes a set of pre-defined available buffer spaces scattered in the routing region. These buffer spaces are represented by small squares, as demonstrated by the dark grey areas b_1 and b_2 in Figure 6 (a). It is assumed that only one buffer can be inserted in each space and the center of the buffer must lie within the square. Larger buffer spaces can easily be expressed as a union of small spaces.

Intuitively, a buffer space is considered for buffer insertion only when a routing path passes through it, since no extra wire cost is incurred under this condition. However, even if no path passes through a buffer space, it may be worthwhile for the wire to make small detour to increase the possibility of exploiting a buffer space. Based on this idea, we define a territory box for an edge as follows:

Definition 3: For an edge (v_i, v_j), its *territory box* is a rectangle specified by lower-left corner point (x_{min}, y_{min}) and upper-right corner point (x_{max}, y_{max}), such that:

$x_{min} = \min(x_i, x_j) - \phi,$

$y_{min} = \min(y_i, y_j) - \phi,$

$x_{max} = \max(x_i, x_j) + \phi,$

$y_{max} = \max(y_i, y_j) + \phi,$

where ϕ is a small amount of offset.

The idea of a territory box is demonstrated by the light grey regions in Figure 6(b). Note that the territory box for the soft edge (v_0, v_3) is larger than for any solid edges between v_0 and v_3. The rule that we will follow is as follows: *a buffer space is considered for buffer insertion in an edge only when there is an overlap between this buffer space and the territory box of this edge.* In the example of Figure 6, buffer space b_1 overlaps with the territory box of edge (v_0, v_3) and b_2 overlaps with the territory box of (v_3, v_2); therefore, we can insert buffers v_4 and v_5 as in Figure 6(c). After the non-Hanan optimization following the buffer insertion, the wire slack in Figure 6 (c) may be removed and the tree shown in Figure 6(d) may be obtained.

This example shows that the use of soft edges can greatly increase the possibility of overlap as

compared to using predetermined L-shaped connection composed of two solid edges.

We consider both inverting and non-inverting type buffers in our work. The inverting type buffer is simply an inverter and the non-inverting type buffer is composed by a pair of cascaded inverters. The inverter model is same as the driver model in section 4.1 and has a medium driver size.

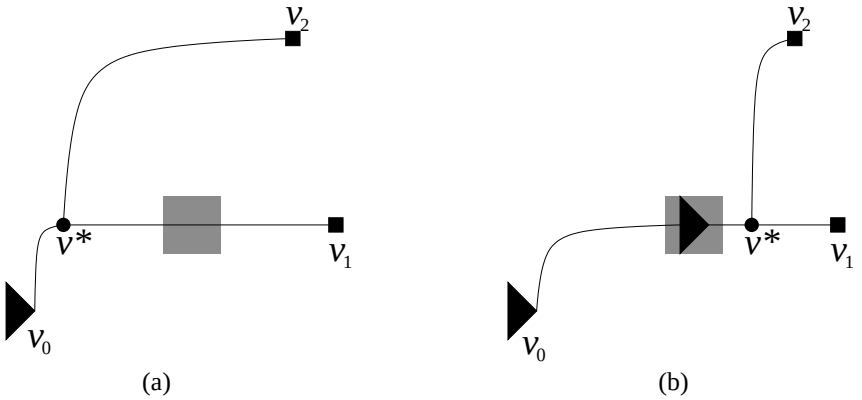

Figure 7: An example that buffer insertion can reduce wire cost further in non-Hanan optimization.

The motivation for combining buffer insertion with non-Hanan optimization can be illustrated by the example in Figure 7. As discussed in section 2.2, in order to reduce wire cost, it is desired to move the connection point as close to CC as possible, i.e., to maximize z. However, the value of z may be capped by the constraint of non-positive delay violation as illustrated in Figure 7(a). The utility of buffer insertion is to relax this timing constraint, if possible, so as to achieve further wire cost reduction as in Figure 7(b).

We state the problem formulation as follows:

Given a source v_0, a set of sinks $V = \{v_1, v_2 ... v_n\}$, timing specifications $Q = \{q_1, q_2, ..., q_n\}$ for all sinks and a set of available buffer spaces $P = \{p_1, p_2, ..., p_m\}$, construct a Steiner routing tree and choose a subset $B_{iv} \subseteq P$ and $B_{ni} \subseteq P$ on which inverting and non-inverting buffers are inserted, respectively, such that the following problem is solved:

$$\begin{array}{cc} \textit{minimize} & \gamma cW + (1-\gamma)(C_{gb} + C_{db})(|B_{iv}| + 2|B_{ni}|) \\ \textit{subject to:} & \max_{v_i \in V}(u_i) \leq 0 \\ \textit{for a specific } \gamma & 0 \leq \gamma \leq 1 \end{array} \tag{2}$$

where C_{gb} and C_{db} are the gate and drain capacitance of an inverting buffer, and γ is the weighting

factor for the wire cost. The purpose of including c, C_{gb} and C_{db} in the objective function is to normalize the wire and the buffer cost into comparable quantities.

4 Problem formulation and properties for FAR-DS

4.1 Problem formulation for FAR-DS

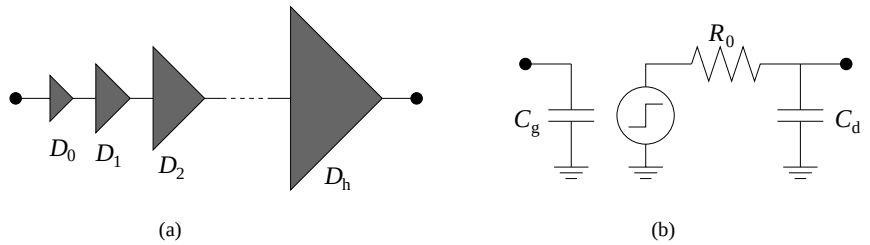

Figure 8: Cascaded drivers and driver model.

We consider the situation where the signal net is driven by a series of cascaded drivers $D_0, D_1, D_2....D_h$ as in Figure 8(a). The driver D_0 is minimum sized and will not be changed in driver sizing. The driver model that we will use is shown in Figure 8 (b). The interconnect delay among these drivers are neglected. The driver sizing problem is to choose optimal number of driver stages h and the proper size for each driver. It is known that for optimal solution, the ratio of driver size at one stage to its previous stage is uniform and no less than 1. We refer to this uniform ratio as the stage ratio ρ.

The objective of FAR-DS is to minimize the cost of the routing tree, subject to a timing constraint at each sink. In contrast with [10], we extend the cost here to include both wire cost and driver cost, i.e., we perform topology optimization and driver sizing simultaneously. The rationale behind this is to permit the driver to share the task of delay optimization with the interconnect by sizing it, thereby obtaining a better result than optimizing the driver size and interconnect topology separately.

We formally state the problem formulation as follows:

Given a source v_0, a set of sinks $V = \{v_1, v_2...v_n\}$, timing specifications $Q = \{q_1, q_2, ...q_n\}$ for all sinks, and stage ratio bound ρ_{max}, construct a Steiner routing tree and find ρ, h such that:

$$\begin{array}{ll} \textit{minimize} & \gamma cW + (1-\gamma)(C_g + C_d)\sum_{j=1}^{h}\rho^j \\ \textit{subject to:} & \max_{v_i \in V}(u_i) \leq 0 \\ \textit{and} & 1 \leq \rho \leq \rho_{max}. \end{array} \tag{3}$$

The second term in the objective function is from the total driver capacitance. The objective function can be interpreted as a minimization of the total wire length and total driver capacitance. The parameter γ is a user-specified weighting factor.

4.2 Properties of solution regions for FAR-DS

For a general connection of a node and its downstream subtree to a partial tree, as illustrated in Figure 2, where a node v_k is to be connected to an edge (v_i, v_j), we investigate the properties of the delay violation function with respect to z and ρ in a two dimensional space. The delay from the cascaded drivers is given by:

$$T_D = hR_0(C_d + \rho C_g) \tag{4}$$

We can combine the interconnect delay with T_D to obtain a general form of the delay violation of any sink u_i as a function of the connection position z and ρ, under the Elmore model as:

$$u_i = f(z, \rho) = -a_2 rcz^2 + \frac{R_0(C_t - cz)}{\rho^h} + a_1 z + R_0 C_g h\rho + a_0 \tag{5}$$

where

$$a_2 = 0 \text{ or } 1, \quad 0 \le z \le CC < \tfrac{C_t}{c}, \quad 1 \le \rho \le \rho_{max}, \tag{6}$$

with a_0 and a_1 being constants. The parameter C_t is the total load capacitance seen by the driver in the last stage when v_k is connected to v_i directly. A derivation of the above results is described in the appendix.

When ρ is fixed, $u_i = f(z)$ is a quadratic function of z and the coefficient of the second order term is non-positive. Therefore we can obtain the following result:

Property 1: $u_i = f(z, \rho)$ is a concave function for a constant value of ρ.

If we keep z constant, there are also properties that will help the search for the optimal solution. These properties can be found by investigating the partial derivatives of u_i with respect to ρ as follows:

$$\frac{\partial u_i}{\partial \rho} = -R_0(C_t - cz)h\rho^{-h-1} + R_0 C_g h \tag{7}$$

$$\frac{\partial^2 u_i}{\partial^2 \rho} = R_0(C_t - cz)h(h+1)\rho^{-h-2} \tag{8}$$

Since $C_t > cz$, $\frac{\partial^2 u_i}{\partial^2 \rho} > 0$ is always true, thus we have the following property:

Property 2: $u_i = f(z, \rho)$ is convex function for a constant value of z.

If we let $\frac{\partial u_i}{\partial \rho} = 0$, we can obtain a curve defined as follows:

$$\rho = \sqrt[h+1]{\frac{C_t - cz}{C_g}} \tag{9}$$

Property 3: $f(z, \rho)$ has minimum value along the curve defined by equation (9) .

This property is especially useful in solution search, since it predicts the bottom of the valley shaped delay violation function surface in the two-dimensional space of z and ρ. One observation is that the curve in equation (9) is independent of which sink is considered, i.e., equation (9) defines the bottom of valley for the delay violation functions of *all* the sinks. We call the curve defined by equation (9) the *valley curve* for delay violations.

In equation (9), when z is at CC, the numerator reaches the minimum and becomes the total load capacitance seen by the driver in the last stage when v_k is connected to CC. Obviously, this total load capacitance is always greater than the minimum gate capacitance, C_g, of a driver. This fact provides the following property:

Property 4: If $0 \leq z \leq CC$, then $\rho = \sqrt[h+1]{\frac{C_t - cz}{C_g}} > 1$.

If we substitute equation (9) into equation (5), we can obtain another important conclusion:

Property 5: $u_i = f(z, \rho)$ is a concave function of z along the curve defined by equation (9).

This valley curve also sets a border for different monotone properties with respect to ρ as follows:

Property 6: For a specific z, $f(z, \rho)$ is a monotone increasing function of ρ when $\rho \geq \sqrt[h+1]{\frac{C_t - cz}{C_g}}$.

Property 7: For a specific z, $f(z, \rho)$ is a monotone decreasing function of ρ when $\rho \leq \sqrt[h+1]{\frac{C_t - cz}{C_g}}$.

These properties are derived from Elmore delays. Though the Elmore delay may have large errors for specific points, its qualitative fidelity is still true [2] and can serve as good strategic guide. Our

experimental results also support this assertion.

5 Algorithms

Algorithm: Non-Hanan_Optimization(T)
Input: Routing tree $T(V, E)$
Output: Optimized routing tree T'
1. $T' = T$
2. Sort all the nodes in descending order of distance to source
3. For each $v_k \in V, k \neq 0$
4. Disjoin v_k and its subtree T_k from T
5. For each edge $(v_i, v_j) \in T \backslash T_k$
6. Reconnect v_k to (v_i, v_j) at optimal location
7. If $\exists$ improvement compared to T'
8. $T' = T$

Figure 9: The non-Hanan optimization algorithm.

Both BINO and FAR-DS consist of two phases. Phase I is the routing tree construction process, which is the same for BINO and FAR-DS. This procedure is called SART (Steiner AWE Routing Tree), and is similar to SERT except that the Elmore model is replaced by a fourth order AWE model.

In SART, starting with a single source, a partial routing tree is grown in a greedy fashion. In each growing step, a previously unconnected sink is selected and connected to an edge in the partial tree such that the maximum delay is minimized.

Phase II of BINO and FAR-DS are different, but share a similarity in the non-Hanan optimization framework. For reference, the outline of the non-Hanan optimization algorithm is shown in Figure 9 [10]. In this framework, all of the sinks are sorted in the descending order of Manhattan distance from source. Then each node v_k and its downstream subtree T_k are disconnected and reconnected back to the routing tree. The routing tree without T_k is represented by $T \backslash T_k$. In the search for the best reconnection point, v_k and T_k are connected to each edge in $T \backslash T_k$ tentatively at the local optimal point. The connection point is selected to be the choice that gives the largest improvement according to the objective.

For two routing trees T_1 and T_2 on the same signal net, if T_1 cannot meet timing constraints and T_2 can result in a smaller maximum delay violation among all sinks, this implies an improvement from T_2 in spite of any cost increase. If T_1 can satisfy all the timing constraints, T_2 provides improvement only

when it reduces cost and satisfies the timing constraints.

Algorithm: BINO_IterativeBufferInsertion	
Input:	SART $T(V, E)$
	Set of buffer spaces P
Output:	Buffered and non-Hanan optimized routing tree T
1.	While $P \neq \emptyset$ and $\exists$ improvement.
2.	For each $p \in P$
3.	For each edge $(v_i, v_j) \in E$
4.	If p overlaps with the territory box of (v_i, v_j)
5.	Insert a buffer into (v_i, v_j) at p tentatively
6.	Assign inverting/non-inverting type $\forall$ buffers $\in T$
7.	Perform non-Hanan optimization for T (Figure 9)
8.	Insert buffer at p_{best}, which gives the largest improvement
9.	$P \leftarrow P - p_{best}$

Figure 10: BINO, iterative buffer insertion algorithm.

In BINO, the non-Hanan optimization framework is embedded in a greedy buffer insertion scheme illustrated by Figure 10. On each buffer space, we insert a buffer tentatively and conduct non-Hanan optimization. After all of the buffer spaces have been tested, the solution that can provide the largest improvement is chosen as the final decision. This process is repeated iteratively until there is no improvement or no buffer space left. The optimal solution of assigning inverting or non-inverting type to each buffer (line 6 in Figure 10) can be achieved through dynamic programming.

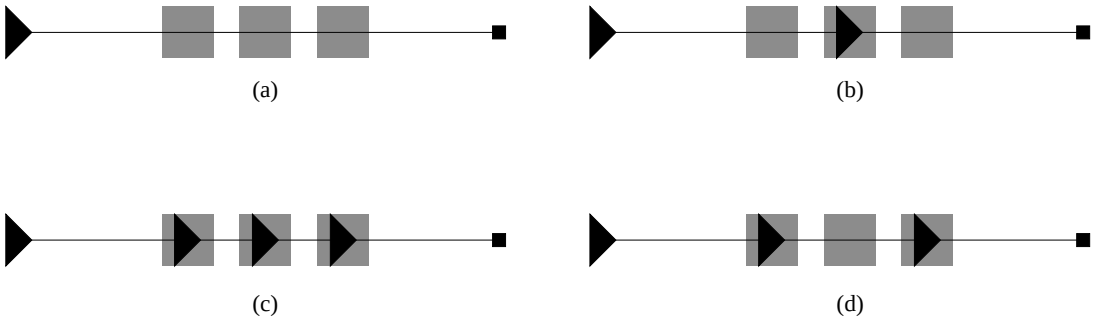

Figure 11: Iterative buffer insertion vs. iterative buffer deletion.

Since we only insert one buffer in each iteration, the ability to obtain an optimal buffer insertion solution is hindered, as shown by the single-sink example in Figure 11. It is well known that optimal buffer locations often distribute evenly along an interconnect path [15]. Therefore, for the net in Figure 11, the optimal solution may be as shown in Figure 11 (d). If we insert only one buffer in an iteration,

the first iteration is likely to result in the scenario shown in Figure 11 (b) and the optimal solution cannot be reached.

In order to alleviate the above difficulty, we supplement the method with an iterative buffer deletion procedure using a method similar to [7], that is described in Figure 12. In this scheme, we first insert buffers at all spaces that overlap with any edges. Then we delete one buffer in each iteration in a greedy fashion similar to iterative buffer insertion. Since this proceeds in the opposite direction as compared to the iterative buffer insertion, it plays a complementary role. For the example in Figure 11, the iterative buffer deletion starts with (c) and can naturally result in the optimal solution in (d). On the other hand, if the optimal solution is (b), iterative buffer deletion is worse than iterative buffer insertion.

In our work, we perform both iterative buffer insertion and iterative buffer deletion independently for a net and choose the better of the two results.

Algorithm: BINO_IterativeBufferDeletion	
Input:	SART $T(V, E)$
	Set of buffer spaces P
Output:	Buffered and non-Hanan optimized routing tree T
1. $B \leftarrow \emptyset$	
2. For each $p \in P$	
3. For each edge $(v_i, v_j) \in E$	
4. If p overlaps with the territory box of (v_i, v_j)	
5. Insert buffer b into (v_i, v_j) at p	
6. $B \leftarrow B \cup b$	
7. Assign inverting/non-inverting type $\forall b \in B$	
8. While $B \neq \emptyset$ and $\exists$ improvement.	
9. For each buffer $b \in B$	
10. Remove b from T tentatively	
11. Assign inverting/non-inverting type $\forall b \in B$	
12. Perform non-Hanan optimization for T (Figure 9)	
13. Remove buffer b_{best}, which gives the largest improvement	
14. $B \leftarrow B - b_{best}$	

Figure 12: BINO, iterative buffer deletion algorithm.

In Phase II of FAR-DS, a two-dimensional search replaces the role of the quasi-binary-search in MVERT, which is line 6 in Figure 9, to find an optimal connection point and driver size simultaneously.

When we reconnect a node v_k to an edge (v_i, v_j), we look for a 3-tuple (z, ρ, h) such that the objective function of problem (3) is minimized while the maximum delay violations for all sinks are non-positive.

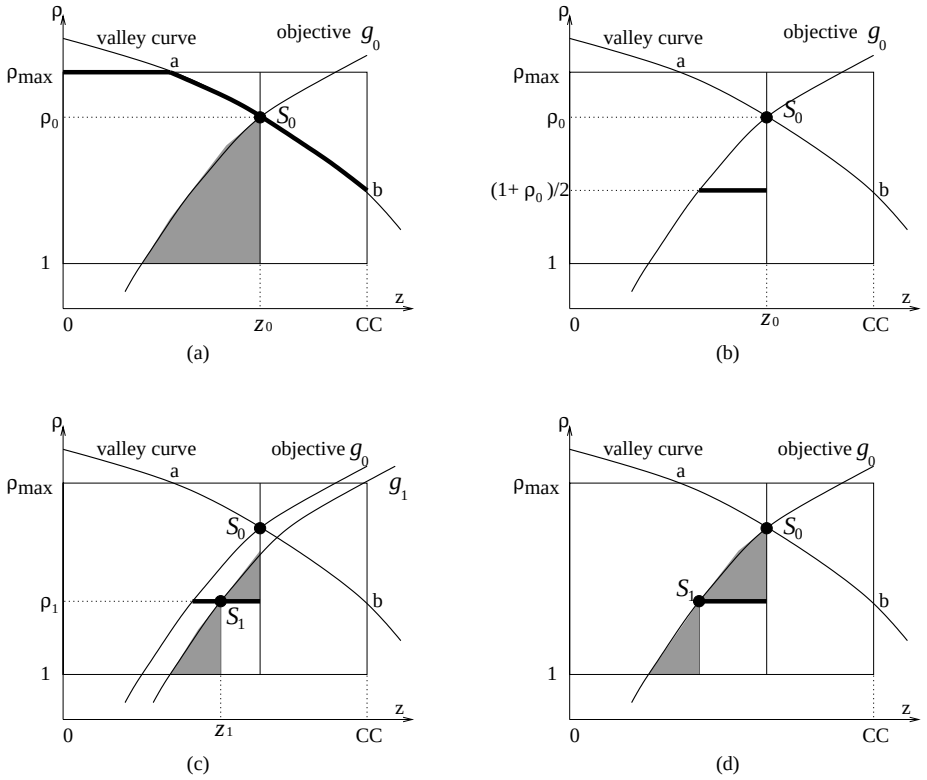

Figure 13: Solution search scheme for FAR-DS.

We vary h between 1 and h_{max} and search an optimal (z, ρ) pair in a two dimensional plane for a fixed h value.

For this case, $cW = C_t - cz$ and the objective (3) can be translated to:

$$\begin{array}{ll} \text{minimize} & g = -\gamma cz + (1-\gamma)(C_g + C_d)\sum_{j=1}^{h} \rho^j \\ \text{subject to:} & \max_{v_i \in V}(u_i) \leq 0 \\ \text{and} & 0 \leq z \leq CC, \;\; 1 \leq \rho \leq \rho_{max}. \end{array} \tag{10}$$

For a specific value of g, the objective function above corresponds a curve in the (z, ρ) plane, as the objective curves shown in Figure 13. The objective (10) can be interpreted as to find a point in (z, ρ) plane such that the constraints in (10) are satisfied at this point and the point is on a objective curve as low as possible.

We will illustrate the optimal solution search scheme through the Figure 13. The solution search can be restricted within the rectangle bounded by $0 \leq z \leq CC$ and $1 \leq \rho \leq \rho_{max}$. Consider the valley curve defined by equation (9). This curve is always above $\rho = 1$ in the interval $0 \leq z \leq CC$, according to

Property 4. One common scenario is that this valley curve intersects with upper border of the rectangle at a point a and with the right border at b, as in Figure 13 (a). From *Property 3* and *Property 7*, we can say that in the rectangle defined above, u_i reaches its minimum on the segment $\rho = \rho_{max}$ to the left of a, and on valley curve specified by equation (9) to the right of a. These two segments can be integrated into a single function:

$$\rho = \min(\rho_{max}, \sqrt[h+1]{\frac{C_t - cz}{C_g}}), 0 \leq z \leq CC \tag{11}$$

which is the thickened line in Figure 13 (a). Note that equation (11) is valid even when the valley curve does not intersect the rectangle, or if the set of points on the segment to the left of a is empty. This function provides us with a convenient way to check for the existence of a solution within the rectangle. From *Property 3* and *Property 7*, if no solution that satisfies all constraints exists on the curve defined by equation (11), then we can say that no solution exists within the rectangle. According to *Property 1* and *Property 5*, u_i is a concave function on the curve (11), both to the left and to the right of point a. Thus, we can apply the quasi-binary-search technique in [10] to search for the rightmost solution on this curve that satisfies all constraints. If such a solution exists, we call it the zero order solution, designated as $S_0(z_0, \rho_0)$ in Figure 13(a).

After the zero order solution has been found, the region can be further refined to search for the optimal solution. This is demonstrated in the shaded region in Figure 13 (a). The region $z > z_0$ can be excluded, since no feasible point exists on the valley curve in this region. An objective function curve is drawn through S_0, which satisfies:

$$g_0 = -\gamma c z_0 + (1-\gamma)(C_g + C_d)\sum_{j=1}^{h} \rho_0^j \tag{12}$$

We can eliminate the region that lies above this curve, because the value of g at all points above this line exceeds g_0. The remainder of the search space is the sector confined by the objective function curve defined by (12), by $z = z_0$ and by $\rho = 1$, which is indicated by the shaded region in Figure 13(a).

The search within this sector also proceeds in a binary search fashion, by starting from the middle segment defined by $\rho_1 = (1 + \rho_0)/2$, which is the thickened segment in Figure 13 (b). On this segment, *Property 1* holds and a quasi-binary-search can again be applied to obtain the rightmost solution on it,

Algorithm: FAR-DS_ReconnectVSearch

Input: Routing tree $T \backslash T_k$
Subtree T_k
Node v_k
Edge (v_i, v_j)

Output: Optimal connection between v_k and (v_i, v_j), ρ and h

1. For $h = 1$; $h \leq h_{max}$; h++;
2. Search solution along valley curve defined by equation(11)
3. If no solution found
4. Return
5. Else
6. $S_0(z_0, \rho_0) \leftarrow$ rightmost solution
7. SearchSector($S_0, 1$)

Function: SearchSector(S_{top}, ρ_{base})

F1. Obtain curve $g_{top} = -\gamma c z_{top} + (1-\gamma)(C_g + C_d) \sum_{j=1}^{h} \rho_{top}^j$
F2. $\rho_{mid} = (\rho_{top} + \rho_{base})/2$
F3. $S_{mid}(z_{mid}, \rho_{mid}) \leftarrow$ intersection between curve g_{top} and $\rho = \rho_{mid}$
F4. Search solution along $\rho = \rho_{mid}$ between z_{mid} and z_{top} by quasi-binary-search
F5. If no solution found
F6. If $\rho_{mid} - \rho_{base} < resolution$
F7. Return no solution
F8. Else
F9. Return $SearchSector(S_{top}, \rho_{mid})$ and $SearchSector(S_{mid}, \rho_{base})$
F10. Else
F11. If $\rho_{mid} - \rho_{base} < resolution$
F12. Return rightmost solution
F13. Else
F14. $S_{mid}(z_{mid}, \rho_{mid}) \leftarrow$ rightmost solution
F15. Obtain curve $g_{mid} = -\gamma c z_{mid} + (1-\gamma)(C_g + C_d) \sum_{j=1}^{h} \rho_{mid}^j$
F16. $S_{top}(z_{top}, \rho_{top}) \leftarrow$ intersection between curve g_{mid} and $z = z_{top}$
F17. Return $SearchSector(S_{top}, \rho_{mid})$ and $SearchSector(S_{mid}, \rho_{base})$

Figure 14: FAR-DS, reconnection and driver sizing in valley-guided search.

namely, $S_1(z_1, \rho_1)$; we refer to this as the first order solution. After the first order solution has been found, the previously described solution refinement technique can be used to obtain two new smaller sectors shown by the shaded regions in Figure 13 (c) where the optimal solution will be searched. Even if there is no solution on this segment, the search region can be refined to the two sectors like in (d). We call this solution search scheme as valley-guided search (V-search), and describe it in Figure 14.

The above is the method to search optimal (z, ρ) for a specific h. The optimal h is found by a sweep from $h = 1$ to $h = h_{max}$ and the above search is carried out for each h value. The value of h_{max} is given

by [26]:

$$h_{max} = \left\lceil \frac{\ln(C_t/C_g)}{\ln \rho^*} \right\rceil, \tag{13}$$

$$\ln \rho^* = 1 + \frac{C_d}{C_g \rho^*} \tag{14}$$

Since the use of valley curve increases the dependency of the solution on the Elmore delay model, and we use a higher order AWE model to evaluate the delays for every sink in our algorithm, it is possible that the discrepancy between Elmore model prediction and the actual AWE evaluation may give rise to a suboptimal solution.

We suggest an alternative search method called the iterative search (I-search) scheme that does not depend on Elmore model quantitatively and illustrate it in Figure 15. In this method, we begin with an initial ρ and perform non-Hanan optimization to obtain an optimal z for this value of ρ. Next, this z is fixed and an optimal ρ is searched and so on. This process is repeated until there is no further improvement. From *Property 6*, we know that the delay violation function u_i is a convex function along ρ direction, thus, we cannot apply the quasi-binary-search suggested by [10] along ρ direction. We perform the search in a manner between binary search and linear search. If the maximum delay violation is non-positive for a specific value of ρ, we continue to search a better solution at a smaller ρ value, otherwise, we must search at both larger and smaller values.

Algorithm: FAR-DS_ReconnectISearch	
Input:	Routing tree $T \backslash T_k$ Subtree T_k Node v_k Edge (v_i, v_j)
Output:	Optimal connection between v_k and (v_i, v_j), ρ and h
1. For $h = 1$; $h \leq h_{max}$; h++	
2. $\rho \leftarrow$ initial guess	
3. While $\exists$ improvement	
4. Search z_{best} which gives best improvement while ρ is fixed	
5. $z \leftarrow z_{best}$	
6. Search ρ_{best} which gives best improvement while z is fixed	
7. $\rho \leftarrow \rho_{best}$	

Figure 15: FAR-DS, reconnection and driver sizing in iterative search.

6 Complexity analysis

From [11], the computation cost for MVERT is $O(n^4 + n^4 \cdot \frac{L}{\epsilon})$. The first term corresponds to the Phase I in MVERT, which is a variation of SERT. The parameter L is the maximum length over all edges and ϵ represents the resolution for the quasi-binary-search in the Phase II of MVERT. Since the quasi-binary-search may fall into a linear search in the worst case, there is no logarithmic term here.

Although we use the fourth order AWE instead of Elmore in BINO, as the number of traversals or iterations is fixed, the complexity for each delay calculation remains $O(n)$. Thus the cost for Phase I (SART) in BINO is $O(n^4)$. In Phase II of BINO, there are two layers of iterations outside of each non-Hanan optimization, each of which is upper-bounded by the number of buffer spaces. The combination of the total cost is $O(m^2 \cdot n^4 \cdot \frac{L}{\epsilon})$. This conclusion is true for both iterative buffer insertion and iterative deletion.

The complexity of FAR-DS is same as MVERT in the outer loops. The difference is at the computation cost of reconnection part (line 6 of Fig. 9), where FAR-DS performs a search in the entire (z, ρ) space. The computation factor from searching along the ρ direction is bounded by $(\rho_{max} - 1)/\tau$, where τ is the resolution on ρ. Since the h value is swept from 1 to h_{max}, the complexity of FAR-DS is $O(h_{max} \cdot n^4 \cdot \frac{L}{\epsilon} \cdot \frac{\rho_{max}}{\tau})$.

The above results only provide a loose bound, because the worst case for the quasi-binary-search along the z direction is almost impossible in practice. Therefore, the computation cost in average case is one order lower than the above theoretic results.

7 Experimental results

The experiments are emphasized to test the improvement from our algorithms in terms of both timing and cost objectives defined in the problem formulation. Each signal net is randomly generated and are tested for BINO and two FAR-DS algorithms, as well as SERT and MEVRT for comparisons. In order to obtain a more general conclusion, we include both IC and MCM technology in the experiments and the number of sinks ranges from 5 to 20. To form a common base for comparisons with FAR-DS, we use cascaded drivers also in SERT, MVERT and BINO, and choose $h = 1$ and $\rho = 2.5$, which can provide a middle level of driving ability. For all of the timing results, driver and wire delays are calculated from RC and a fourth order AWE model, respectively.

The experimental results are shown in Table 2 and Table 3 for IC and MCM technology, respectively. The parameters for MCM are from [2]. The IC parameters correspond to 0.18 μm technology and are scaled from the data in [2]. The buffer space locations for BINO are generated randomly. The area of each buffer space is chosen to be $100\mu m \times 100\mu m$ for IC and $200\mu m \times 200\mu m$ for MCM. According to our experiments, the variations of delay from the change of a buffer position within a buffer space is small and can be neglected. The offset ϕ for the territory box of an edge is set to be half of the buffer space size. In experiment for FAR-DS, the value of ρ_{max} was chosen as 4 for both IC and MCM technology. Since we consider the situation where the interconnect resources are more stringent, the weighting factor for wire cost is chosen to be 0.7 for both BINO and FAR-DS.

In Table 2 and Table 3, the left-most column is the number of sinks for each test. The parameter W is the total wire length and the u_{max} is the maximum delay violation. The column labeled m corresponds to the number of input buffer spaces, and to its left is the number of buffers, $|B|$, finally inserted. The CPU time in seconds for BINO and FAR-DS are also listed. The last row provides the percentage change of total wire length compared to the result of SERT.

Since the timing constraints are quite stringent, most of the maximum delay violations, u_{max}, from the results of SERT are positive. Sometimes MVERT even results in a worse delay violation than SERT, due to errors from the Elmore delay model. In other cases, the improvements from MVERT on both delay and wire cost are limited and the timing constraints are often unsatisfied, because the specification is unachievable without driver sizing or buffer insertion. This hinders the ability of pure non-Hanan optimization to reduce the cost further and BINO or FAR-DS becomes a necessary step. Both BINO and FAR-DS can also satisfy the timing constraints that are impossible for SERT and MVERT.

Besides timing improvement, we can see that BINO and FAR-DS can reduce significantly more cost than MVERT under these somewhat harsh conditions. Sometimes MVERT may even increase the wire length to meet the timing constraint which can be seen from the result of the first net in IC technology. The results from the two different variants of FAR-DS have no significant difference.

Comparing the experimental results from BINO and FAR-DS, we can see that BINO can provide more wire cost reduction than FAR-DS in most cases, and the larger timing slacks from BINO also indicate its potential on dealing with even more stringent timing constraints. Although FAR-DS is not so powerful as BINO, it shows an adaptive nature that can often trade off the timing slack into less

driver cost. This is especially true for the *V-search* scheme of FAR-DS, whose most timing slacks are close to zero.

These experiments were carried out on a SUN Ultra-10 station. The computation time of FAR-DS mostly depends on the size of signal nets while the CPU time of BINO is more irregular because it also depends on the number of buffer spaces overlapping with routing tree. In most cases, the CPU time is within one minute. In the worst case for a net of 20 sinks, the run time is less than four minutes for both FAR-DS and BINO. On the whole, the computational cost of our algorithm is reasonable, since these optimizations are carried out only for global timing-critical nets.

We also perform experiments to check the effect from using soft edges on the same set of nets and the result is shown in Table 4. This result confirms the conclusion that using soft edges can greatly increase the possibility that buffer spaces overlaps with routing edges.

Table 2: Experimental results on $.18\mu m$ IC, $h = 1, \rho = 2.5$ for SERT, MVERT and BINO

	SERT		MVERT		BINO				FAR-DS: I-search					FAR-DS: V-search				
n	W	u_{max}	W	u_{max}	W	u_{max}	$\|B\|/m$	cpu	W	u_{max}	ρ	h	cpu	W	u_{max}	ρ	h	cpu
5	226	4.51	243	-1.19	189	-9.48	3/6	9	187	0	2.9	3	2	187	0	2.9	3	2
5	219	-1.26	166	1.31	153	-2.78	1/7	6	155	-1.70	4.0	1	2	155	-0.02	2.5	1	3
5	292	-2.83	280	-1.03	231	-5.31	1/8	3	235	-0.01	2.6	3	3	235	-0.07	2.7	3	3
10	374	-0.82	341	-0.54	280	-1.69	2/8	24	295	-0.07	1.8	3	13	282	-0.41	1.9	3	19
10	310	-0.45	257	1.48	249	-2.87	1/7	14	252	-0.02	2.9	1	19	252	-0.33	3.1	1	25
10	401	1.92	381	0.91	348	-4.32	1/8	33	349	-1.85	4.0	1	14	349	-0.02	3.1	1	19
15	462	7.57	383	5.94	367	-2.28	2/9	61	370	-0.21	1.7	3	35	370	-0.72	1.8	3	49
15	626	4.37	555	-0.25	470	-0.27	2/10	86	494	0	2.1	5	64	494	-0.09	3.6	3	86
20	595	10.84	537	8.26	469	-0.02	3/10	204	488	-0.01	3.4	3	119	495	-0.43	1.9	3	166
20	564	9.05	481	4.88	433	-1.60	2/11	88	452	-0.06	1.9	3	79	452	-0.47	1.9	3	104
Ave $\Delta W(\%)$			-10		-22				-20					-20				

Table 3: Experimental results on MCM, $h = 1, \rho = 2.5$ for SERT, MVERT and BINO

	SERT		MVERT		BINO				FAR-DS: I-search					FAR-DS: V-search				
n	W	u_{max}	W	u_{max}	W	u_{max}	$\|B\|/m$	cpu	W	u_{max}	ρ	h	cpu	W	u_{max}	ρ	h	cpu
5	444	1.82	427	0.37	332	0	1/6	12	340	0	1.7	3	3	347	-0.14	1.8	3	4
5	429	1.21	410	1.17	332	-0.71	1/7	5	376	-0.02	1.8	3	3	357	-0.07	4.0	1	5
5	478	-0.57	454	0.17	381	-4.91	1/6	3	412	0	2.2	3	3	405	0	2.5	3	5
10	617	4.14	539	3.59	480	-2.10	1/10	38	506	-0.04	1.9	3	16	506	-0.11	3.3	1	19
10	624	-0.55	505	1.08	465	-0.96	2/7	9	482	-0.14	4.0	1	17	482	-0.01	3.8	1	24
10	618	0.99	519	-0.07	486	-2.84	1/8	34	480	-2.01	4.0	1	14	480	-0.61	2.7	1	18
15	810	3.80	755	2.43	652	-0.64	2/10	81	706	-0.04	4.0	1	43	698	0	3.4	1	63
15	797	-0.99	695	0.37	656	-0.45	1/9	78	660	-0.01	3.7	1	42	660	-0.01	3.9	1	53
20	1253	4.69	1149	4.06	900	-0.30	2/12	204	962	-0.12	1.9	3	121	979	-0.36	2.1	3	153
20	1025	3.65	883	3.13	791	-1.91	2/8	105	826	-0.03	3.7	1	84	862	-0.10	2.7	3	124
Ave $\Delta W(\%)$			-10		-23				-19					-19				

Table 4: Comparison on number of overlaps between buffer spaces and routing edges with and without using soft edges

# nets	# overlaps w/o soft edge	# overlaps w soft edge
20	34	71

8 Conclusion

When we extend the non-Hanan optimization to improve the performance of critical nets where both timing and wire resources are stringent, buffer insertion is shown to be a strong augmentation to the timing optimization toolkit, even with location restrictions. A combination of driver sizing and non-Hanan optimization can provide a continuous two-dimensional space. A search for the optimum in this space may be guided by properties derived from the Elmore delay model, which may have large quantitative errors but good qualitative fidelity. These properties are used to direct heuristics that use a fourth order AWE model for wire delay calculation. For drivers, a more accurate model can be applied in place of an RC switch model in a similar fashion. Experimental results show that both BINO and FAR-DS can bring both timing and wire cost improvements significantly.

As compromises to computation time, uniform driver stage ratio and RC driver model are adopted in this work. In future work, we would like to replace them with non-uniform stage ratio and a more elaborated driver model, respectively. Also an efficient integration of BINO and FAR-DS will be considered.

9 Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments.

Appendix

Similar to the work of [2], a delay violation function model with respect to connection location and driver stage ratio under the Elmore delay is derived as follows.

For a general routing tree depicted in Figure 2, we wish to connect node v_k and its downstream subtree T_k to edge (v_i, v_j) in tree $T \backslash T_k$ at a point within the bounding box defined by v_i and CC. We use C_j and C_k to represent the load capacitance seen from node v_j and v_k, respectively. The resistance of the driver that drives the routing tree directly is expressed as R_d.

First we consider the delay function with respect to only the connection location which is the Man-

hattan distance z, from v_i to the connection point v'.

If a node is not in the downstream of node v_i, its Elmore delay from source is as follows:

$$f_1 = R_d(C_t - cz) + \lambda_0 + \lambda_1(l_{ik} - z), \tag{15}$$

where λ_0 and λ_1 are constants. Recall that C_t is the total load capacitance seen from the last stage of driver if v_k is connected to v_i.

The Elmore delay from v_i to v' is given by:

$$f' = rcz(\frac{z}{2} + l_{ij} - z + l_{ik} - z) + rz(C_j + C_k). \tag{16}$$

From v' to any node in T_j, the delay can be obtained as:

$$f_2 = r(l_{ij} - z)(\frac{c(l_{ij} - z)}{2} + C_j) + \lambda_2. \tag{17}$$

Similarly, the delay from v' to any node in T_k is:

$$f_3 = r(l_{ik} - z)(\frac{c(l_{ik} - z)}{2} + C_k) + \lambda_3. \tag{18}$$

Both λ_2 and λ_3 are constants. If a sink is in T_j, its Elmore delay is form of the sum of f_1, f' and f_2. When a sink is in T_k, its Elmore delay is the sum of f_1, f' and f_3. If a sink is not in the downstream of n_i, its Elmore delay is simply f_1. In all these cases, the delay is either a linear or a quadratic function of z with non-positive coefficient for the second order term. Therefore, we can obtain the conclusion that delay or delay violation function for any sink is a concave function with respect to z, which is same as *Theorem 1*.

For cascaded drivers, $R_d = \frac{R_0}{\rho^h}$, and we need to take into account the driver delay $T_D = hR_0(C_d + \rho C_g)$. After we combine the interconnect with driver delay, we can obtain a general form of delay violation function vs. z and ρ as follows:

$$f(z, \rho) = -a_2 rcz^2 + \frac{R_0(C_t - cz)}{\rho^h} + a_1 z + R_0 C_g h\rho + a_0 \tag{19}$$

where a_0 and a_1 are constants, and $a_2 = 0$ or 1. This is the same as equation (5).

References

[1] J. Lillis and P. Buch, "Table-lookup methods for improved performance-driven routing," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 368–373, 1998.

[2] K. D. Boese, A. B. Kahng, B. A. McCoy and G. Robins, "Near-optimal critical sink routing tree constructions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, No. 12, pp. 1417-36, Dec. 1995.

[3] J. Lillis, C. K. Cheng, T. T. Lin and C. Y. Ho, "New performance driven routing techniques with explicit area/delay tradeoff and simultaneous wire sizing," *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, pp. 395-400, Jun. 1996.

[4] A. Vittal and M. Marek-Sadowska, "Minimum delay interconnect design using alphabetic trees," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 392–396, 1994.

[5] J. Cong and C. K. Koh, "Interconnect layout optimization under higher-order RLC model," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 713-720, 1997.

[6] F. J. Liu, J. Lillis and C. K. Cheng, "Design and implementation of a global router based on a new layout-driven timing model with three poles," *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1548-51, 1997.

[7] J. Cong and B. Preas, "A new algorithm for standard cell global routing," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 176-179, 1988.

[8] J. Cong and C. K. Koh, "Simultaneous driver and wire sizing for performance and power optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 2, No. 4, pp. 408-425, Dec. 1994.

[9] S. S. Sapatnekar, "RC interconnect optimization under the Elmore delay model," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 392-396, 1994.

[10] H. Hou and S. S. Sapatnekar, "Routing tree topology construction to meet interconnect timing constraints", *Proceedings of the ACM International Symposium on Physical Design*, pp. 205-210, 1998.

[11] H. Hou, J. Hu and S. S. Sapatnekar, "Non-Hanan routing", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, No. 4, pp. 436-444, April, 1999.

[12] L. P. V. Ginneken, "Buffer placement in distributed RC-tree networks for minimal Elmore delay", *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 865-868, 1990.

[13] J. Lillis, C. K. Cheng and T. Y. Lin, "Simultaneous routing and buffer insertion for high performance interconnect", *Proceedings of the Sixth Great Lakes Symposium on VLSI*, pp. 148-153, 1996.

[14] J. C. Shah and S. S. Sapatnekar, "Wiresizing with buffer placement and sizing for power-delay tradeoffs", *Proceedings of the International Conference on VLSI Design*, pp. 346-351, 1996.

[15] C. C. N. Chu and D. F. Wong, "Closed form solution to simultaneous buffer insertion/sizing and wire sizing", *Proceedings of the ACM International Symposium on Physical Design*, pp. 192-197, 1997.

[16] T. Okamoto and J. Cong, "Buffered Steiner tree construction with wire sizing for interconnect layout optimization", *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 44-49, 1996.

[17] A. Salek, J. Lou and M. Pedram, "A simultaneous routing tree construction and fanout optimization algorithm", *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 625-630, 1998.

[18] H. Zhou, D. F. Wong, I-M. Liu and A. Aziz, "Simultaneous routing and buffer insertion with restrictions on buffer Locations", *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 96-99, 1999.

[19] W. C. Elmore, "The transient response of damped linear network with particular regard to wideband amplifiers," *Journal of Applied Physics*, Vol. 19, pp. 55-63, 1948.

[20] J. Rubinstein, P. Penfield and M. A. Horowitz, "Signal delay in RC tree networks," *IEEE Transactions on Computer-Aided Design*, Vol. CAD-2, No. 3, pp. 202-211, July 1983.

[21] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 9, No. 4, pp. 352-366, Apr. 1990.

[22] C. L. Ratzlaff, N. Gopal and L. T. Pillage, "RICE: Rapid interconnect circuit evaluator," *Proc. 28th ACM/IEEE Design Automation Conference*, pp. 555-560, 1991.

[23] J. Qian, S. Pullela and L. T. Pillage, "Modeling the effective capacitance for the RC interconnect of CMOS gates," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 12, pp. 1526-35, Dec. 1994.

[24] R. Gupta, B. Krauter, B. Tutuianu, J. Willis and L. T. Pileggi, "The Elmore delay as a bound for RC trees with generalized input signals," *Proc. 33rd ACM/IEEE Design Automation Conference*, pp. 364-369, 1995.

[25] D. G. Luenberger, "Linear and Nonlinear Programming," *Addison-Wesley Publishing Company, Reading, Massachusetts.*, 1984.

[26] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: a Systems Perspective*, 2nd edition, Reading, MA, Addison-Wesley, pp. 229-231, 1993.