Maze Routing with Buffer Insertion and Wiresizing *

Minghorng Lai
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712

D.F. Wong
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712

Abstract

We propose an elegant formulation of the Maze Routing with Buffer Insertion and Wiresizing problem as a graph-theoretic shortest path problem. This formulation provides time and space performance improvements over previously proposed dynamic-programming based techniques. Routing constraints such as wiring obstacles and restrictions on buffer locations and types are easily incorporated in the formulation. Furthermore, efficient software routines solving shortest path problems in existing graph application libraries can be applied. We construct a BP-Graph such that the length of every path in this graph is equal to the Elmore delay. Therefore, finding the minimum Elmore delay path becomes a finite shortest path problem. The buffer choices and insertion locations are represented as the vertices in the BP-Graph. The interconnect wires are sized by constructing a look-up table for buffer-to-buffer wiresizing solutions. We also provide a technique that is able to tremendously improve the runtime. Experiments show improvements over previously proposed methods.

1 Introduction

Routing has become one of the most challenging tasks facing VLSI circuit designers as fabrication technology moves into deep sub-micron territory. It has been shown that, as a result of scaling, interconnect delays will become the most significant part of total system delay[1, 8]. Routing information is highly desirable at an early stage during the layout design process for helping designers achieve various performance optimizations. Furthermore, with rapidly increasing number of transistors on chips, routing is likely to become a computationally expensive stage in the system design process.

Maze routing has been a successful method for global routing tasks. The goal of maze routing is to find a route to connect two terminals in a routing area, often represented as a grid graph. When there are no restrictions on the buffer locations in the routing area, it has been shown that the buffered minimum Elmore delay path between source s and sink t can be found by first finding the shortest path from s to t and then inserting buffers and sizing wires on this path. However, preplaced macro blocks in the routing area could prevent buffers from being inserted at certain locations in the routing area. In this case, a shortest path is not necessarily a minimum Elmore delay path because the macro blocks could restrict buffer locations on the shortest path[9]. In [9] a dynamic-programming based technique was devised to find the buffered minimum Elmore delay path in the grid graph with restrictions on buffer locations. Although the technique is able to insert buffers and find the minimum delay path in a fairly small amount of time, its performance degrades rapidly if the interconnect wires in the routing area are to be sized at the same.

In this paper, we present a formulation of the Maze Routing with Buffer Insertion and Wiresizing problem as a shortest path problem. Previously, dynamic programming based methods have gained popularity with various buffer insertion and wiresizing problems; however, dynamic programming has several drawbacks such as huge memory requirements. Our formulation is more natural, intuitive and efficient; it provides improvements, both in time and space, over previously proposed dynamic programming based techniques. We focus on the possible buffer locations on the routing grid graph and construct a BP-Graph(Buffer Planning Graph) in which the vertices represent possible buffer choices at different buffer locations. A look-up table is constructed for sizing wires along the path between two buffers at different nodes. Our algorithm finds the minimum delay path, inserts buffers and performs wiresizing on the path at the same time. Routing constraints such as wiring obstacles and restrictions on buffer types and locations are naturally incorporated in the formulation. Experiments show improvements in both runtime and space over previously proposed methods.

The remainder of the paper is organized as follows: In Section 2, we briefly review the Elmore delay model and formally define the Maze Routing with Buffer Insertion and Wiresizing problem. In Section 3, we propose our formulation of the Maze Routing with Buffer Insertion and Wiresizing problem as a regular shortest path problem. The time and space complexity of the new method will be analyzed. We also propose a technique that can further speed up the algorithm. In Section 4, we show our experiments and results, and provide some concluding remarks.

2 Delay Model and Problem Definition

In this section we briefly review the Elmore delay model and formally define the Maze Routing with Buffer Insertion and Wiresizing problem.

2.1 Delay Model

Elmore delay model has been a popular delay model since it was proposed in [3] because of its simple analytical form, fidelity and other properties[5]. If we model a wire segment as

*This work was partially supported by a grant from the Intel Corporation and by the Texas Advanced Research Program under Grant No. 003658288.

DAC 2000, Los Angeles, California

a π-type element (Fig. 1), the Elmore delay of the individual wire segment is:

$$t_{wire} = r_w(\frac{1}{2}c_w + c_d),$$

where c_d is the downstream capacitance for this wire segment. Similarly, for an inserted buffer with capacitance c_b, resistance r_b, and intrinsic delay t_{in}, the Elmore delay is computed as:

$$t_{buffer} = r_b c_d + t_{in}.$$

The total Elmore delay along any interconnect path is computed by summing together the individual wire delays and buffer delays. We will use the Elmore delay model in this paper.

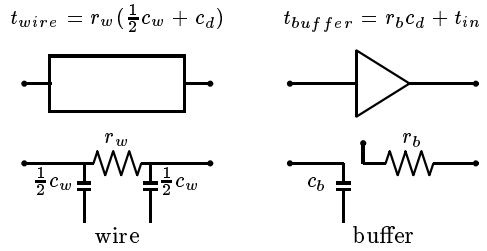


Figure 1: Elmore Delay Model.

2.2 Problem Definition

The goal of maze routing is to find a route between two terminals in a routing area, which is often represented as a grid graph $G = (V, E)$. Some wiring obstacles and restrictions on buffer locations and types may be present in the routing area. Each edge (u, v) in E is considered to represent one wire segment. Wire library W contains wire choices for sizing the wire segments in G. Buffers are chosen from library B to be inserted at the grid nodes in G. A sample routing grid graph and a buffered minimum delay path is in Fig. 2.

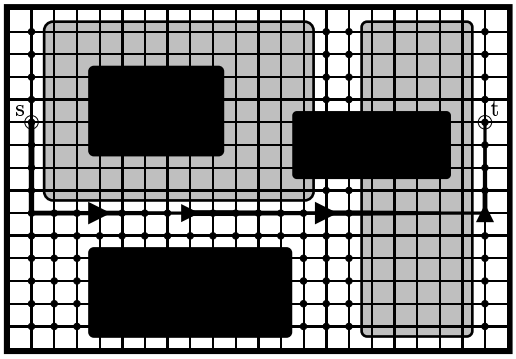


Figure 2: A Routing Grid Graph and a Buffered Minimum Elmore Delay Path. The dark areas represent wiring obstacles. No buffers can be inserted in gray areas and dark areas. Wires, however, are allowed to go through gray areas. The possible buffer locations are denoted by the solid circles at some of the grid line intersections.

The Maze Routing with Buffer Insertion and Wiresizing problem can be formally described as follows:

DP-Routing for Buffer Insertion and Wiresizing

$Q \leftarrow \{(C_L, 0, -1, t)\}$;
while(Q `not empty`) **do**
  $(c, t, b, u) \leftarrow$ `extract-min(`Q`)`;
  if $c = 0$ **then**
    `return` t;
  if $u = s$ **then**
    $S \leftarrow (0, t + R_D c, b, u)$;
    `add` S `to` Q `and prune`;
  else
    for each $(u, v) \in E$ **do**
      for w `= 0 to` $W - 1$
        $c' \leftarrow c + c_w(u, v)$;
        $t' \leftarrow t + \frac{1}{2} r_w(u, v)(c_w(u, v) + c)$;
        $S \leftarrow (c', t', -1, v)$;
        `add` S `to` Q `and prune`;
  if $p(u) = 1$
    for b' `= 0 to` $B - 1$
      $S \leftarrow (c_{b'}, t + r_{b'} c + d_{b'}, b', u)$;
      `add` S `to` Q `and prune`;

Figure 3: Dynamic Programming for Mazing Routing with Buffer Insertion and Wiresizing

Problem 1 (Maze Routing with Buffer Insertion and Wiresizing) *Given a routing grid graph $G = (V, E)$, a buffer library B, a buffer function p with $p(v) = 1$ indicating buffer insertion allowed at node v, a wire library W, two nodes $s, t \in V$, find a buffered path $P = (v_1, v_2, \ldots, v_n)$, with $v_1 = s$, $v_n = t$, $b(v_i) \in B \cup \{-1\}$ where $b(v_i) = -1$ indicates that no buffer is inserted at v_i, and $w(v_j, v_{j+1}) \in W$, such that the Elmore delay for path P is minimized.*

We notice that without restrictions on buffer locations this problem can be easily solved by finding the shortest path between source s and sink t, and then inserting buffers and sizing wires on the path to minimize the delay. However, with restrictions on buffer locations, the shortest path is not guaranteed to be the minimum delay path.

3 Shortest Path Formulation

In this section we present our formulation of the Maze Routing with Buffer Insertion and Wiresizing problem as a shortest path problem and describe an algorithm for solving this problem. The running time and space requirement of the algorithm are analyzed. We also propose a simple technique to speed up the algorithm.

3.1 Previously Proposed Methods

Previously, Zhou et. al. [9] proposed a dynamic programming based approach for finding buffered minimum delay path in the grid graph with restrictions on buffer locations; however, they didn't consider wiresizing. Although their method performs well for finding buffered minimum delay path with buffer location restrictions, when wiresizing is taken into account the performance degrades considerably.

Dynamic programming has been a popular method for various buffer insertion and wiresizing problems [4,6,7]. The key procedure of dynamic programming based methods is the bottom-up, recursive construction of capacitance-delay

pairs (c,t) – c is the downstream capacitance and t is the delay. These (c,t) pairs are constructed at each node in the graph and propagated upstream from sink to source. Let R_D be the driver resistance and C_L be the load capacitance. Initially, there is only one (c,t) pair at the sink with $c = C_L$ and $t = 0$. For each (c,t) at node u, a new (c',t') is constructed for each of the neighbors of u. Wiresizing is performed by choosing a wire type from a library W. Buffer insertion is also considered for minimizing the delay. A key observation was made in [4] that (c_1,t_1) is redundant and can be safely discarded if there is a (c_2,t_2) such that $c_1 \geq c_2$ and $t_1 \geq t_2$, because c_1, compared to c_2, will always incur a greater delay upstream, and add this delay to the already greater $t1$. This observation allows pruning of many (c,t) pairs and keeps the set from growing too large. The set of (c_i,t_i) pairs at the source s is finally used to select a solution with minimum delay $R_D * c_i + t_i$. The pseudo code for the dynamic programming based method [9] is in Fig. 3.

3.2 Shortest Path Formulation

We will show our formulation of the Maze Routing with Buffer Insertion and Wiresizing problem by an example. Fig. 4 is a simple routing grid graph $G = (V,E)$. The goal is to find a minimum delay path from source s to sink t and to insert buffers and size interconnect wires simultaneously. No wires are allowed in the dark area as it represents a wiring obstacle. Wires are allowed to go through the buffer obstacle area(gray area); however, no buffers can be inserted in this area. The are 6 possible buffer locations in this grid graph. For simplicity, we will consider only one buffer type b_0 for this example.

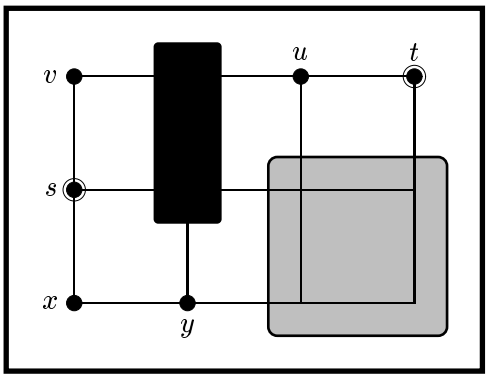


Figure 4: A Simple Routing Grid Graph G.

We proceed to construct our BP-Graph $BG = (V_{BG}, E_{BG})$ as follows: First, the vertices in our BP-Graph represent possible buffer choices and locations in the graph grid G. For each node u in grid graph G, we begin by creating a new vertex in BP-Graph u_{b_i} for each buffer type b_i allowed to be inserted at u. If a buffer type b_j is not allowed at grid node u, vertex u_{b_j} is not created. In graph BG in Fig. 5, each vertex v_{b_i} represents a placement of buffer b_i at node v in graph G. The weight of edge (u_{b_i}, v_{b_j}) in BG is the minimum delay between node u and v in grid graph G, with buffer b_i inserted at node u and buffer b_j inserted at node v, respectively, and no other buffers inserted along the path from node u to node v. The minimum delay between u_{b_i} and v_{b_j} is computed easily because it is a simple function of

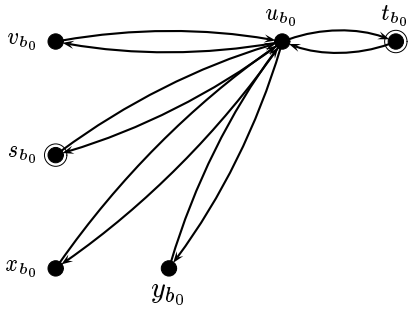
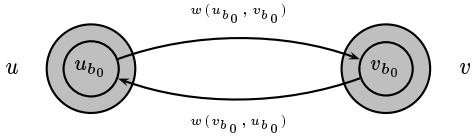
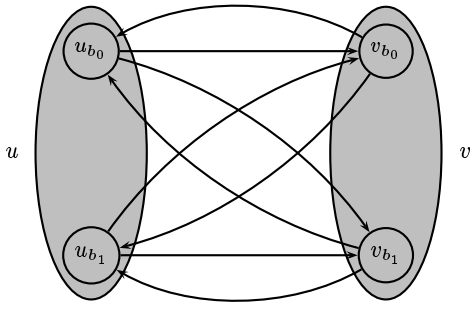


Figure 5: Vertex u_{b_0} and Its Connections to Neighbors in the BP-Graph BG for Grid Graph G in Fig. 4.

the shortest distance between u and v in G, the resistance of b_i, the capacitance of b_j, and the wire library W. For example, $w(u_{b_0}, v_{b_0})$ in BG is equal to the minimum delay from u to v in G with buffer b_0 inserted at both u and v. A look-up table stores the buffer-to-buffer wiresizing solutions. The weights between each pair of two vertices u_{b_i} and v_{b_j} are computed from the table according to buffer type b_i, b_j, and the shortest distance between u and v in G. Fig. 5 shows the vertex u_{b_0} and its connections to all its neighbors in BP-Graph BG. Notice that no nodes in buffer obstacle area or wire obstacle area are included in this graph. The problem of finding a minimum delay path is then simply equivalent to finding a shortest path from s to t in BG. We can use any shortest path algorithm to find the corresponding minimum delay path in G.



(a) Single Buffer Type



(b) Multiple Buffer Types

Figure 6: Construction of BP-Graph with Multiple Buffer Types.

Fig. 6 shows the construction of BP-Graph when B has multiple buffer types. At node u, for each buffer type b_i, a vertex u_{b_i} is created and connected to all neighboring vertices v_{b_j}, if u and v are connected in G.

SP-Routing for Buffer Insertion and Wiresizing

```
for each pair of nodes u, v ∈ V do
    compute shortest distance d(u,v) in G
for each u ∈ V do
    for each b_i ∈ B do
        if b_i allowed at u
            create a new vertex u_{b_i}
for each pair of u_{b_i}, v_{b_j}
    if  d(u,v) < ∞ then
        use (b_i, b_j, d(u,v)) as index to check
        look-up table for wiresizing solution and
        b_i-to-b_j minimum delay min_d(i,j,d(u,v));
        w(u_{b_i}, v_{b_j}) = min_d(i,j,d(u,v));
create s_{b_-1} for no buffer insertion at s
create t_{b_-1} for no buffer insertion at t
computer w(s_{b_-1}, t_{b_-1})
for each u_{b_i}
    use d(s,u) and look-up table to
    compute w(s_{b_-1}, u_{b_i})
for each u_{b_i}
    use d(u,t) and look-up table to
    compute w(u_{b_i}, t_{b_-1})
Find the Shorest Path from s_{b_-1} to t_{b_-1}.
```

Figure 7: Using Shortest Path Formulation to Solve Mazing Routing with Buffer Insertion and Wiresizing

Without wiresizing, the weight of edge (u_{b_i}, v_{b_j}) in BP-Graph is simply a function of the shortest distance between node u, v in the grid graph G, the buffer capacitance of b_j, and resistance of b_i. However, with wiresizing, some computation is needed to produce the wiresizing solutions for the shortest path from node u to v. Notice that this path can go through the buffer obstacle areas because there are no buffered inserted on the path from u to v. Techniques such as quadratic programming proposed in [2] or dynamic programming [4,6,7] are suitable for sizing wires between pairs of buffers. We use a look-up table to store the results to avoid recomputation of the same buffer-to-buffer wiresizing solutions. This look-up table can also be re-utilized for multi-net routing. Our SP-Routing for Buffer Insertion and Wiresizing algorithm is in Fig. 7.

Fig. 8 shows an example of finding the buffered minimum delay path by using the Shortest Path Formulation of the Mazing Routing with Buffer Insertion and Wiresizing problem to solve a Shortest Path problem on BP-Graph and find the shortest path in Fig. 9. The buffers on the buffered minimum delay path are indicated by the labels of the vertices in the shortest path $P1$. The wiresizing solutions are obtained from the look-up table.

The following theorem proves that Mazing Routing with Buffer Insertion and Wiresizing problem is a Shortest Path problem:

Theorem 1 *Maze Routing with Buffer Insertion and Wiresizing Is Shortest Path Problem.*

Consider a buffered minimum delay path $P = (v_1, v_2, \ldots, v_n)$, with $v_1 = s$ and $v_n = t$ in grid graph G. Assume that buffers are inserted at nodes $v_{l_1}, v_{l_2}, \ldots v_{l_m}$ on the path. It is then obvious that there is a corresponding

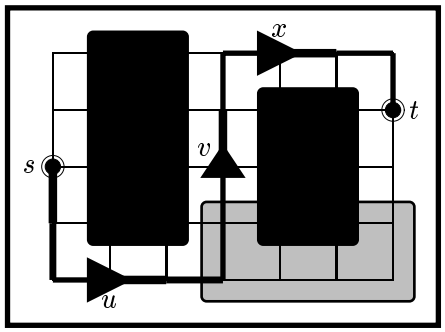


Figure 8: Finding The Buffered Minimum Delay Path by Using the Shortest Path Formulation of the Mazing Routing with Buffer Insertion and Wiresizing Problem to Solve a Shortest Path Problem on BP-Graph and Find the Shortest Path in Fig. 9.

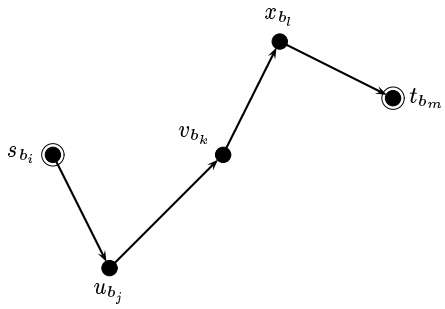


Figure 9: A Shortest Path $P1$ in BG-Graph. The corresponding buffered minimum delay path is in Fig. 8.

path $P' = (s, v_{b_1}, \ldots, v_{b_m}, t)$ in BG. Since P is the minimum delay path in G, P' is the shortest path from s to t in BG.

The following two theorems establish the time and space complexity of the Shortest Path Formulation of Maze Routing with Buffer Insertion and Wiresizing Algorithm:

Theorem 2 (Time Complexity of Shortest Path Formulation of Maze Routing with Buffer Insertion and Wiresizing) *The running time of the Maze Routing for Buffer Insertion and Wiresizing Algorithm is* $O(|V|^2 \log(|V|))$.

For a grid graph $G = (V, E)$, at most $|B||V|$ new vertices are created in the BP-Graph $BG = (V_{BG}, E_{BG})$. It takes $O(|V|^2 \log(|V|))$ to compute the shortest distances between all pairs of nodes in G. The runtime for computing the weights in BG is $O(|V_{BG}|^2)$. The runtime of the dynamic programming based technique depends on the maximum number of sub-solutions, ct_{max}, at each node. When wiresizing is considered, the number of sub-solutions grows rapidly along the path.

Theorem 3 (Space Complexity of Shortest Path Formulation of Maze Routing with Buffer Insertion and Wiresizing) *The space complexity of the Simultaneous Maze Routing and Buffer Insertion Algorithm is* $O(|B|^2|V|^2)$.

	DP-Routing		SP-Routing	
name	Memory (Mb)	Time (s)	Memory (Mb)	Time (s)
SRL1	2.88	741	0.524	35.3
SRL2	2.71	919	0.318	18.2
SRL3	2.70	827	0.355	19.5
SRL4	3.09	1044	0.740	51.0
SRL5	3.08	1306	0.672	60.2
SRL6	2.69	961	0.572	38.1
SRL7	2.82	969	0.767	46.1
SRL8	2.88	767	0.384	25.5
SRL9	2.85	868	0.479	22.6
SRL10	3.48	1243	0.869	71.0

Table 1: SP-Routing vs. DP-Routing.

	DP-Routing		SP-Routing	
name	Memory (Mb)	Time (s)	Memory (Mb)	Time (s)
TRL1	1.70	292	0.190	11.3
TRL2	1.66	387	0.126	5.1
TRL3	1.61	322	0.151	6.1
TRL4	1.88	432	0.327	15.2
TRL5	2.01	522	0.355	18.2
TRL6	1.72	406	0.250	13.4
TRL7	1.77	420	0.384	17.2
TRL8	1.68	337	0.110	5.6
TRL9	1.72	393	0.226	9.6
TRL10	1.66	374	0.197	7.3

Table 2: SP-Routing vs. DP-Routing.

Graph BG is a dense graph with $O(|B||V|)$ vertices. The space complexity for dynamic programming based technique is $O(|V|ct_{max})$. When the number of wire choices is increased, dynamic programming could require a huge amount of space to store the sub-solutions.

A simple observation is made to reduce the size of V_{BG}. First, the shortest distance d_{st} between s and t is computed. We can use this shortest distance to limit the number of vertices in V_{BG} by observing that a node u is not on the minimum delay path from s to t if $d_{su} + d_{ut} > d_{st}$. Since the size of V_{BG} is one of the deciding factors for the performance of the Shortest Path Formulation, this is very helpful for improving both runtime and space requirements.

4 Experimental Results and Concluding Remarks

We tested our formulation and algorithm on a set of routing areas. Routing obstacles and macro blocks are randomly generated and placed in these routing areas. We used the technology parameters in [8]. The driver resistance is set to be 100Ω, the load capacitance $1fF$, respectively. We test our method with four buffer types with capacitance c_b ranging from $0.13fF$ to $9.92fF$, and resistance r_b ranging from 1990Ω to 90830Ω. There are five wire choices in our wire library W with $1.3\Omega < r_w < 121.1\Omega$ and $1.72fF < c_w < 55.13fF$. Our experiments were conducted on a Pentium Pro machine with 64 megabytes of memory.

We compare our SP-Routing algorithm with original DP-Routing algorithm. Table 1 shows the results for 10 different routing areas with 3-8 obstacles and macro blocks. Both methods are used to route from randomly generated source s to sink t and insert buffers and size interconnect wires at the same time. Dynamic programming is used to construct the look-up tables. The technique mentioned in Section 3 for reducing the size of BP-Graph is implemented in our program. Our method consistently outperforms the original DP-Routing algorithms. The look-up tables for buffer-to-buffer delays for these test cases are constructed within less than 30 seconds. With wiresizing, the sizes of the sub-solution sets for dynamic programming increase significantly. Furthermore, the same sub-solutions are scattered among different nodes in the grid graph, increasing both runtime and space usage. In contrast, our SP-Routing algorithm uses a look-up table to avoid storing identical sub-solutions, and is more efficient than dynamic programming.

Another set of results can be found in Table 2. Again our method is able to compute the exact routing, buffer insertion and wire sizing solutions faster than the original DP-Routing algorithm.

The shortest-path formulation also allows consideration of congestion avoidance in the routing area by assigning weights to the vertices representing buffer locations. This is also helpful for buffer zone planning. The lookup-table construction only needs to be done once and can be re-used in multi-net maze routing.

References

[1] H. B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley, Reading, Mass., 1990.

[2] C. Chu and D. F. Wong, " A New Approach to Simultaneous Buffer Insertion and Wire Sizing," *IEEE Trans. on CAD*, 1997.

[3] W. C. Elmore, "The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifiers," *J. Applied Physics*, 1948, pp. 55-63.

[4] L. P. P. P. van Ginneken, "Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay," *Proc. Int'l Symp. on Circuits and Systems*, 1990, pp. 865-868.

[5] R. Gupta et al., "The Elmore Delay as a Bound for RC Trees with Generalized Input Signals," *Proc. ACM/IEEE Design Automation Conference*, 1995, pp. 364-369.

[6] J. Lillis, C. -K. Cheng, and T. -T. Y. Lin, "Optimal and Efficient Buffer Insertion and Wire Sizing," *IEEE Custom Integrated Circuits Conf.*, 1995, pp. 259-262.

[7] J. Lillis, C. -K. Cheng, and T. -T. Y. Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model," *Proc. Int'l Conf. on Computer-Aided Design*, Nov. 1995, pp. 138-143.

[8] *The National Technology Roadmap for Semiconductors*, Semiconductor Industry Association, 1997.

[9] H. Zhou, D. F. Wong, and I. Liu, "Simultaneous Routing and Buffer Insertion with Restrictions on Buffer Locations," *Proc. ACM/IEEE Design Automation Conference*, 1999.