A Graph Based Algorithm for Optimal Buffer Insertion Under Accurate Delay Models *

Youxin Gao
Avant! Corporation
46871 Bayside Parkway
Fremont, CA 94538
Tel: +1-510-413-7170
Fax: +1-510-413-8080
e-mail: gao@avanticorp.com

D.F. Wong
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
Tel: +1-512-471-9527
Fax: +1-512-471-8885
e-mail wong@cs.utexas.edu

Abstract

Buffer insertion is an efficient technique in interconnect optimization. This paper presents a graph based algorithm for optimal buffer insertion under accurate delay models. In our algorithm, a signal is accurately represented by a finite ramp which is characterized by two parameters, shift time and transition time. Any accurate delay model, such as delay models based on the transmission line model and SPICE simulations, can be incorporated into our algorithm. The algorithm determines the optimal number of buffers and their locations on a wire such that some optimization objective is satisfied. Two typical examples of such optimization objectives are minimizing the 50% threshold delay and minimizing the transition time. Both can be easily determined in our algorithm. We show that the buffer insertion problem can be reduced to a shortest path problem. The algorithm can be easily extended for simultaneous buffer insertion and wire-sizing, and complexity is still polynomial. The algorithm can also be extended to deal with problems such as buffer insertion subject to transition time constraints at any position along the wire.

1. Introduction

As the VLSI technology shrinks down to nanometer range, interconnect delay becomes the bottleneck in achieving high performance circuits. Techniques which are aiming at reducing interconnect delay are necessary and important. These techniques, including buffer insertion, wire-sizing and simultaneous buffer insertion and wire-sizing, have been extensively studied in recent years. By inserting buffers in a wire, not only delay can be greatly reduced, but also the output waveform can be improved in terms of the reduced transition time. Under the Elmore delay model and a linear gate model, some efficient algorithms for simultaneous buffer insertion and wire-sizing have been proposed in [6, 5]. van Ginneken in [15] proposes a dynamic programming based algorithm for the optimal buffer insertion. The algorithm has also been extended for simultaneous buffer insertion and wire-sizing in [11], and noise avoidance in [2, 4]. However, it is well known that both the Elmore delay model and the linear gate model are inaccurate. Furthermore, both models are not aware of the input waveforms, which is becoming increasingly important in today's deep sub-micron design. Therefore the optimal solution under these simple models may be inferior [1]. Only recently, the authors in [1, 12] extend van Ginneken's algorithm by using both accurate interconnect and gate delay models. In calculating delay for a wire, both use the moment matching technique based on a lumped circuit approximation. However, in [1], the signal waveform is not actually taken into account. The authors assume a fixed input slope in calculating delay for a buffer. In [12], the signal waveform is considered in buffer insertion. But its pruning process, which is similar to the one in Ginneken's algorithm, only allows one waveform to survive to the next stage. As we already know, a signal waveform is characterized by two parameters, the shift time and transition time. If two waveforms have different shift times and transition times, it is hard to tell which one is better than the other (except for some special cases, e.g., same transition time but different shift times). Therefore some potential non-inferior waveforms can be removed by this pruning process. Furthermore, the lumped circuit approximation presented in [1, 12] is not as accurate as the transmission line model or simulations based on SPICE especially in high frequency [17].

In this paper, we present a graph based algorithm for optimal buffer insertion under accurate delay models. In our algorithm, a signal is accurately represented by a finite ramp which is characterized by two parameters, shift time and transition time. Any accurate delay model, such as delay models based on the transmission line model and SPICE simulations, can be incorporated into our algorithm. The algorithm determines the optimal number of buffers and their locations on a wire such that some optimization objective is satisfied. Two typical examples of such optimization objectives are minimizing the 50% threshold delay and minimizing the transition time. Both can be easily determined in our algorithm. We show that the buffer insertion problem can be reduced to a shortest path problem. The algorithm can be easily extended for simultaneous buffer insertion and wire-sizing, and complexity is still polynomial. The algorithm can also be extended to deal with problems such as buffer insertion subject to transition time constraints at any position along the wire.

Comparing with previous approaches on buffer insertion, our algorithm has the following advantages:

1. Our graph based algorithm is simpler and easier to understand and implement. The optimal solution can be found through the shortest path algorithm.
2. In our algorithm, a signal is accurately represented by a finite ramp. Any accurate delay model such as the transmission line model and the model based on SPICE simulations can be used. This makes our algorithm general of use.
3. The traditional optimization objective in buffer insertion only aims at reducing the signal delay. Our algorithm can not only deal with this objective, but also deal with some other objectives, such as optimal buffer insertion for minimmum transition time, and/or subject to transition time constraints at any position. Both are shown as shortest path problems.

The rest of the paper is organized as follows. Section 2 introduces an analytical delay model used in our algorithm. In section 3, we show that the optimal buffer insertion is a shortest path problem. Section 4 presents an efficient algorithm which combines graph construction with the shortest path algorithm. In section 5, we present some experimental results.

2. Accurate Delay Models

Since a single delay value (e.g., 50% delay) in a general delay model can not fully characterize the waveform of a signal, we approximate each signal as a finite ramp which is characterized by two parameters, shift time S and transition time T (see Figure 1). Any arbitrary signal is represented by a finite ramp by connecting two points at 10% and 90% threshold voltages,

*This work was partially supported by the National Science Foundation under grant CCR-9912390, by the Texas Advanced Research Program under Grant No. 003658288, and by grants from Avant!, Intel and IBM.

respectively. The usual 50% delay is thus $S + \frac{1}{2}T$. To simplify the notation, we use a pair (S, T) to represent a finite ramp. In the rest of the paper, we show how to use an analytical delay model, where wire delay is based on the transmission line model and buffer delay is characterized by k-factor equations.

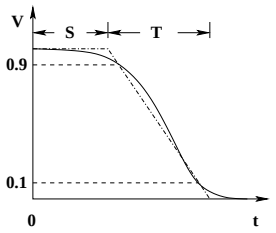

Figure 1. An arbitrary signal is represented by a finite ramp with shift time S and transition time T.

In buffer insertion, a typical wire-buffer system is shown in Figure 2(a), where a wire is connecting two buffers. In Figure 2(b), the buffer is represented by a circuit which contains a input capacitance C_B and a voltage source V_B. The waveform calculation is cascaded in terms of a pair (S, T) through each wire-buffer system.

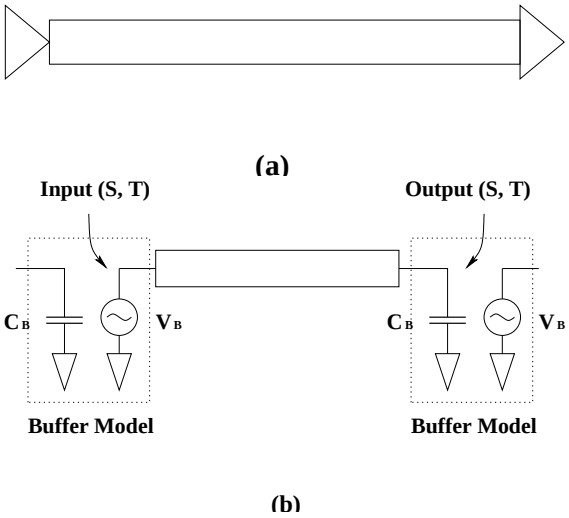

Figure 2. (a). A wire connecting two buffers. (b). Signal waveform calculation is cascaded in terms of a pair (S, T).

We use the transmission model to model the interconnect wire, which is found more accurate than methods based on lumped circuit approximation [9, 17]. Under the transmission line model, voltage and current at any position are described by the telegraph's equations. In modeling the interconnect wire, we take both fringing capacitance and inductance into consideration. Both effects are found important in today's design [3, 8]. The waveform calculation for a wire consists of the following steps (details can be found in [8, 9]):

1. Use transmission line model to derive $\mathcal{ABCD}$ parameters for an interconnect wire by solving telegraph's equations.
2. Delay model is based on three pole approximation.
3. Obtain analytical forms for calculating the first three terms b_1, b_2 and b_3 in the transfer function $H(s) = \frac{1}{1+b_1 s+b_2 s^2+b_3 s^3+\cdots}$, and derive the time domain response by assuming a finite ramp input.
4. Use analytical delay expressions to calculate delay at any threshold voltage [9, 10], as well as the output waveform (S_{out}, T_{out}).

To calculate the output waveform (S_{out}, T_{out}) for a buffer, we make use of k-factor equations. Delays at threshold voltages 10% and 90% are expressed in terms of the following empirical equations [16]:

$$T_{10} = (k_1 + k_2 C_w)T_{in} + k_3 C_w^2 + k_4 C_w + k_5 \quad (1)$$

$$T_{90} = (k'_1 + k'_2 C_w)T_{in} + k'_3 C_w^2 + k'_4 C_w + k'_5 \quad (2)$$

where T_{in} is the transition time of an input slope, $C_w = C_L/w$, C_L is the buffer's load capacitance, and w is the buffer's channel width. Therefore the waveform of the voltage source V_B can be calculated in terms of a pair (S, T) as: $S = (9T_{10} - T_{90})/8$, $T = (T_{90} - T_{10})/0.8$ for a rising ramp; and $S = (9T_{90} - T_{10})/8$, $T = (T_{10} - T_{90})/0.8$ for a falling ramp. Without loss of generality, throughout the rest of the paper we assume that PMOS and NMOS have the same driving capability. Therefore it is not necessary to distinguish between rising transition and falling transition. Since k-factor equations need a load capacitance, the interconnect wire which is connected to the buffer has to be approximated by a load capacitance. Because of resistance shielding, the buffer can not see the total capacitance of its down stream interconnects [14]. Therefore using total capacitance is not a good approximation. A widely used method is to calculate an effective capacitance for the wire, and connect it as a load to the buffer. The calculation consists of following steps (details see [9]):

1. For the wire which is connected to the buffer as a load, use analytical expressions to calculate the input admittance $Y(s)$.
2. Expand $Y(s)$ into Taylor series and keep first three terms, i.e., $Y(s) = y_1 s + y_2 s^2 + y_3 s^3$.
3. Use the technique in [13] to determine an equivalent CRC Π-model which matches $Y(s)$.
4. Calculate the effective capacitance C_{eff} for this CRC $\Pi-$model using technique in [14].

For the delay model we described, we have the following observation.

Observation 1 Shift time S is additive, i.e., if an input waveform $(0, T_{in})$ causes an output waveform (S_{out}, T_{out}), then another input waveform (S_{in}, T_{in}) will cause an output waveform $(S_{in} + S_{out}, T_{out})$.

For the buffer macro-model in equations (1-2), this is obvious, since delays are not depending on input shift time. In the delay calculation for a wire, suppose the final voltage response is: $V_{out}(s) = V_{in}(s)H(s)$, where $V_{in}(s)$ is the input voltage response, and $H(s)$ is the transfer function. Assume the corresponding voltage response in the time domain is $v_{out}(t)$. If the input signal is shifted by $\mathcal{S}$, the new input signal will be $V'_{in}(s) = e^{-s\mathcal{S}}V_{in}(s)$. The corresponding output voltage is then $V'_{out}(s) = e^{-s\mathcal{S}}V_{in}(s)H(s) = e^{-s\mathcal{S}}V_{out}(s)$. The new voltage response in the time domain is thus the original voltage response $v_{out}(t)$ shifted by $\mathcal{S}$. □

A direct result of this observation is that we can specify an input waveform as $(0, T_{in})$ instead of (S_{in}, T_{in}) for either a buffer or a wire, or a system contains both. This is especially useful when we use SPICE simulations in our buffer insertion algorithm. For the wire-buffer system shown in Figure 2(a), the output waveform is uniquely determined by the wire length and the input waveform (thus T_{in} only). Because of buffer insertion, wires are divided into smaller segments, where each segment is connecting two buffers. Suppose a wire has L_{max} possible buffer locations which are uniformly distributed. For any possible buffer insertion scheme, the wire length of any segment can be any from 1 to $L_{max} + 1$ in some units. Therefore we can build a lookup table for each wire-buffer system of length from 1 to $L_{max} + 1$. The input to the lookup table is a set of transition times $\{(T_{in})_1, (T_{in})_2, \cdots\}$, and the output is a set of pairs $\{(S_{out}, T_{out})_1, (S_{out}, T_{out})_2, \cdots\}$. For any possible wire connection encountered in buffer insertion, given an input waveform $(0, T_{in})$ we can easily determine its output waveform by table lookup.

3. Buffer Insertion is a Shortest Path Problem

The problem we want to solve can be stated as follows:

> **Given**: a wire of length L, width W, driver resistance R_D, and load capacitance C_L.
> **Determine**: the optimal number of buffers and their positions on the wire such that delay through the wire is minimized.

We claim that this problem can be formulated as a shortest path problem, therefore the optimal solution can be determined through the shortest path algorithm. Although we will use accurate delay models for buffer insertion, to illustrate this claim, we first start from the simple Elmore delay model. Later on in this section, we show that the claim is true under accurate delay models.

3.1. Buffer Insertion under the Elmore Delay Model

For a wire shown in Figure 3(a), it has three possible buffer locations. Buffers can be inserted at these locations depending on whether it helps to reduce delay. We will determine the optimal buffer insertion through a graph.

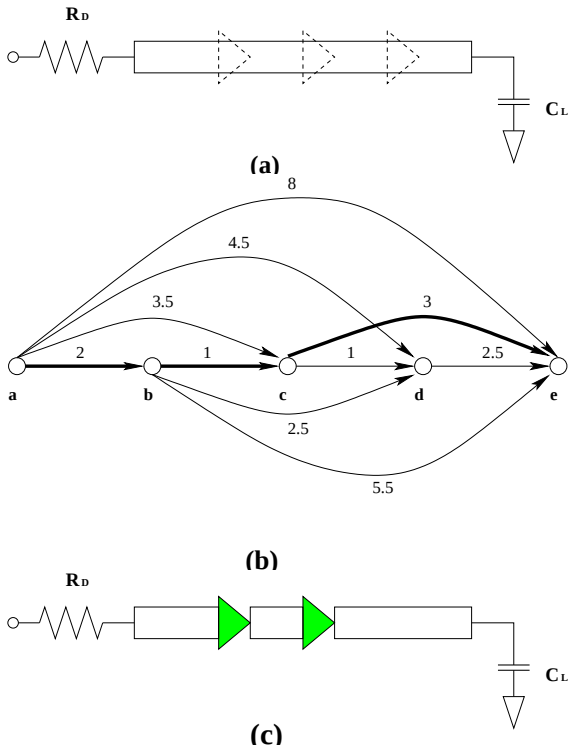

Figure 3. (a). A wire has three buffer locations. (b). Buffer insertion is a shortest path problem. Weights on edges represent delays. The highlighted path **a-b-c-e** is the shortest path. (c). Buffer insertion scheme corresponds to the shortest path **a-b-c-e**.

As shown in Figure 3(b), we construct a graph where **a** is the source node which represents the driver end, **e** is the sink node represents the load end. Nodes **b, c, d** represent three possible buffer locations. Starting from left to right, we connect the current node to all the nodes on its right by directed edges. Each edge corresponds to a possible wire connection between two buffers (hereafter, for notational convenience, we treat both driver and load as buffers). For each edge we are going to connect, its wire length, driver and load are known, so it is straight forward to calculate its delay using the Elmore delay model. We assign the delay value as a weight to the edge. For example, the edge connecting node **a** and **c** has weight 3.5, which means the delay for the wire connecting from the source to the second buffer is 3.5 units. After all edges are assigned with weights, the graph construction is complete. We therefore obtain a direct acyclic graph (DAG).

In the DAG, each path from the source node **a** to the sink node **e** represents a buffer insertion scheme. For example, the highlighted path **a-b-c-e** represents a buffer insertion scheme shown in Figure 3(c), where two buffers are inserted into first two buffer locations, and there is no buffer at the third position. By summing up all the edge weights along the path, each path is thus associated with a path delay, which represents the delay of the correspondent buffer insertion scheme. Among all paths, the path **a-b-c-e** has the shortest delay. Therefore the buffer insertion scheme represented by the path **a-b-c-e** is the optimal solution.

3.2. Buffer Insertion under Accurate Delay Models

Under accurate delay models, signals can not be fully characterized by a single delay value like in the Elmore delay model. As we mentioned in section 2., a signal can be accurately represented by a finite ramp (S, T). Therefore those edge weights shown in Figure 3(b) are no longer meaningful under accurate delay models. For the buffer insertion in Figure 3(a), we will construct a similar but more complicated graph and show that the buffer insertion problem can still be described as a shortest path problem.

For notational convenience, we use level 0 to represent the driver position, and level 4 to represent the load position. Levels 1 to 3 represent three possible buffer locations. Level 5 represents the position where we insert a sink node **f**. There will be more than one node at each level. Note that if there is a wire connecting from a node at level l_1 to l_2, the wire length is $l_2 - l_1$.

Before we construct the DAG, we assume there are T_{max} number of transition time bins TIME[1..T_{max}] which are numbered from 1 to T_{max}. TIME[1] represents the fastest transition time, and TIME[T_{max}] represents the slowest transition time. All these transition times are in ascending order. We then create T_{max} nodes at each level from 0 to 4, and these nodes at the same level are numbered from 1 to T_{max}. Each node with index i at a certain level is associated with an output transition time for a possible connection from a node on its left to this level. The output transition time of such connection is TIME[i]. Furthermore, TIME[i] also serves as the input transition time of a connection from this level to a node on its right.

We construct the graph in the topological order. First we assume a input transition time T_{in} for the source. We then pick a node i as the source node at level 0 such that TIME[i] is the closest to T_{in}. Assume there exist connections from this node to a node at each level on its right. We can calculate the output waveform (S_{out}, T_{out}) for each assumed connection (not yet connected), since the wire length and input waveform $(0, T_{in})$ are known. The calculation can be done either through the analytical delay model or through the lookup table based on SPICE simulations (see section 2.). Then we make the real connection from the source node to a node at each level. The node index k at each level is determined such that T_{out} matches TIME[k]. The edge weight is assigned with S_{out}. By repeating a similar process to nodes level by level, all the nodes except for those at level 4 can be connected to some nodes on its right levels. In Figure 4, all nodes which have been connected through edges are shown as small circles, and those nodes which are not connected by any edges are shown as dots. Finally, we connect all small circles at level 4 (nodes **e1, e2, ..., e7**) to the sink node **f**. Since each circle really represents a transition time, we can assign each edge with a weight which is a half of the transition time. We thus have a DAG shown in Figure 4.

Note that except for those edges connecting from level 4 to level 5 where the edge weights are half of the transition times, all the weights represent shift times. By summing up all the weights along a path from the source to the sink, we will obtain a delay value associated with this path, since delay is equal to the shift time plus a half of the transition time. For example, the highlighted path **a-b1-c1-e2-f** in Figure 4 has delay

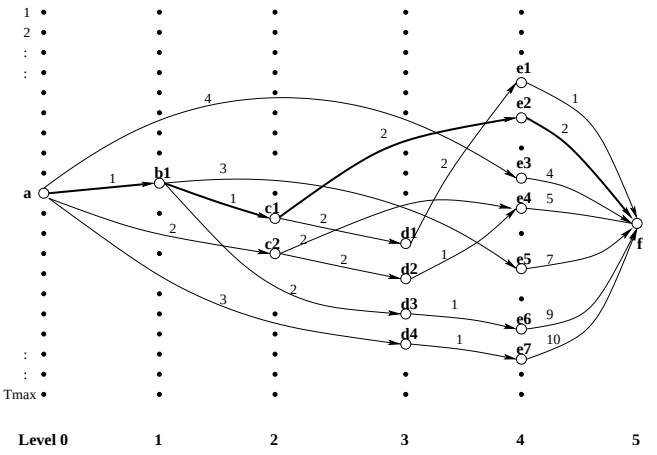

Figure 4. Direct acyclic graph for buffer insertion under accurate delay models. Except for these edges connecting from level 4 to 5 whose weights denote half of transition times, all edge weights represent shift times.

6, which is the shortest path in the graph. Therefore its correspondent buffer insertion scheme (for illustration purpose, it is the same as the one shown in Figure 3(c)) is the optimal solution. Note that there is no guarantee that this buffer insertion scheme happens to have the fastest transition. In fact, a different path **a-b1-c1-d1-e1-f** has the fastest transition time.

Comparing two DAGs shown in Figures 3 (denote it as DAG_Elmore) and Figure 4 (denote it as DAG_Accurate), respectively, they have the following different features.

1. DAG_Accurate has a lot more nodes than DAG_Elmore, but not all of them are connected with other nodes. In DAG_Elmore, all nodes are connected with edges.
2. In DAG_Accurate, edge weights represent shift times or transition times, where as in DAG_Elmore, edge weights represent delays.

Despite these difference, both graphs can be used to determine the optimal solution through the shortest path algorithm.

Remark 1 So far the optimization objective for buffer insertion is to minimize delay. In fact since our approach has taken both transition time and shift time into consideration, it is easy to consider some other optimization objectives. We can minimize the final transition time (i.e., require a fastest transition). This can be also shown as a shortest path problem. We need modify the graph in Figure 4 such that only these edges from level 4 to level 5 are assigned with weights (half of the transition times), and all other edges are assigned with weights 0. The shortest path is thus the path **a-b1-c1-d1-e1-f** (see Figure 4). It corresponds to a buffer insertion scheme where buffers are inserted at all possible locations.

Remark 2 It is also easy to consider a constraint on the final transition time. For example, for the problem in Figure 3(a), we require that the final transition time should be less than 14 units. In this case, nodes **e6** and **e7** at level 4 which have transition times greater than 14 are not allowed in the solution path. We assign ∞ as weights for edges connecting from **e6** and **e7** to **f**. In this example, the shortest path is still **a-b1-c1-e2-f**. In fact, the transition time constraint can be considered anywhere in the graph.

4. Efficient Algorithm for Buffer Insertion

We can follow the idea in the previous section by constructing the graph in topological order, then using the shortest path algorithm to find the optimal solution. However, as we have already seen in Figure 4, lots of nodes are wasted in this way, since they will not be connected to any other nodes. Those unconnected nodes will waste lots of memory. In this section, we present an efficient algorithm, where nodes are created only when they are needed. At the same time, the shortest path algorithm is implicitly implemented in our algorithm. After the graph is constructed, we only have to look at these nodes at level $L_{max}+1$ and choose one according to the optimization objective we use. The shortest path can then be obtained by backtracking. A pseudo-code of our algorithm is shown in Figure 5.

Input: L_{max}- # of buffer locations
TIME[$1..T_{max}$]- transition time bins
Output: $S[0..L_{max}+1]$- sets of nodes
Algorithm **BUILD_GRAPH**
1. let all sets $S[0..L_{max}+1]$ be empty
2. create source node n_0
3. $S[0] = \{n_0\}$
4. for $i = 0$ to $L_{max}+1$
5. while ($S[i] \neq 0$)
6. pick a node $n \in S[i]$
7. $S[i] = S[i] - n$
8. let l be level of node n
9. let (s_{in}, t_{in}) be waveform of n
10. for $k = l$ to $L_{max}+1$
11. compute output (s_{out}, t_{out}) for a wire of length $k-l$ and input $(0, t_{in})$
12. if($\exists x \in S[k]$ && $x.(t_x) = t_{out}$)
13. if($s_{out} + s_{in} < x.(s_x)$)
14. $x.(s_x) = s_{out} + s_{in}$
15. let n be predecessor of x
16. else
17. create node y of level k and output $(s_{out} + s_{in}, t_{out})$
18. let n be predecessor of y
19. $S[k] = S[k] \cup y$

Figure 5. Pseudo-code of Algorithm **BUILD_GRAPH**.

In algorithm **BUILD_GRAPH**, a node contains information such as its level, the predecessor node and waveform (S, T), where S is the shift time, and TIME[T] is the transition time. The waveform of a node n can be extracted as $n.(S,T)$, or $n.(S)$ and $n.(T)$. The running time complexity of our algorithm is $O(L_{max}^2 T_{max}^2)$.

Remark 3 In lines 12-15, if there are two edges connecting to the same node (i.e., these two waveforms have the same transition time), we only keep the one which has faster shift time. The one which has slower shift time will be removed or not be connected, since it definitely causes inferior waveform in later stages. As a result, we can keep as many as possible potential good connections, and this is a better improvement than [12]. Moreover, by doing this, the backtracking path is uniquely determined in our algorithm.

Remark 4 It is easy to extend our algorithm for simultaneous buffer insertion and wire-sizing. For wire-sizing, we specify a width library from which a width is chosen for each wire segment. For example, the width library has N different choices $\{W_1, W_2, ...W_N\}$. To accommodate wire-sizing, we have the following modifications.

1. Every connection in the graph has a width which can be chosen from library $\{W_1, W_2, ...W_N\}$.
2. For each wire-buffer system in Figure 2(a), given the wire length, width and an input waveform $(0, T_{in})$, we can compute the output waveform (S_{out}, T_{out}). This can be done using either the analytical delay model or a lookup table based on SPICE simulations.
3. Modify the algorithm **BUILD_GRAPH** by adding a loop between step 10 and 11 as follows.

10. for $k = l$ to $L_{max}+1$
11. for $w = 1$ to N
12. compute output (s_{out}, t_{out}) for a wire of length $k-l$, width W_w and input $(0, t_{in})$

By considering wire-sizing, the graph will be more complicated, since each edge in the original graph will be split into N different edges. However, the shortest path still gives the optimal solution. The running time complexity of the algorithm for simultaneous buffer insertion and wire-sizing is $O(L_{max}^2 T_{max}^2 N^2)$.

By using the similar idea, we can extend our algorithm further to considering buffer sizing with simultaneous buffer insertion and wire-sizing. The running time complexity is still polynomial.

5. Experimental Results

In this section, we present some experimental results on buffer insertion. The wire parameters are chosen as follows: wire width $w = 1.035\mu m$, unit square resistance $r_0 = 0.092\Omega/\Box$, unit area capacitance $c_0 = 0.03205fF/\mu m$, driver resistance $R_D = 28.3\Omega$, load capacitance $C_L = 0.016pF$, wire length $L = 16,000\mu m$, unit length fringing capacitance $c_f = 0.0877fF/\mu m$, and unit length self inductance $l_0 = 0.73913pH/\mu m$. For buffers, we choose fixed channel width and length for the NMOS transistor as $5\mu m$ and $0.5\mu m$, respectively. For such a long wire, we assume it has L_{max} number of locations where we can insert buffers. We choose $T_{max} = 200$, i.e., there are 200 transition time bins. For the input signal, we specify a ramp by choosing $S = 0$, $T = 1ps$. Throughout our experiments, we use the analytical delay models outlined in section 2.. Without inserting buffers, the wire has delay $T_{50\%} = 1.6179ns$ and transition time $T = 3.4298ns$.

In the following experiments, we study two different optimization objectives. One is to minimize 50% delay (i.e., shortest delay), and the other is to minimize the transition time (i.e., fastest transition). Both shift time and transition time are measured at the final load end.

L_{max}	#	$T_{50\%}$ (ns)	T (ns)	Running Time(s)
7	2	1.2601	1.1394	0.01
15	2	1.2028	1.2008	0.03
31	2	1.1981	0.9734	0.15
63	3	1.1666	0.9237	1.06
127	3	1.1444	0.9237	5.90

Table 1. Optimal buffer insertion to minimize 50% delay.

For the results in Table 1, we determine the optimal buffer insertion such that delay is minimized. As L_{max} increases, the optimal solution seems converge, and the optimal number of buffers is 3. The optimal buffer insertion scheme for $L_{max} = 127$ is shown in Figure 6(a). The running times shown in Table 1 do not include the time to build a lookup table. However, since we use the analytical delay model outlined in section 2., building a lookup table is very fast. For $L_{max} = 127$ which is the most time consuming experiment, the running time is about $6s$. Notice that all problems can be solved within a few seconds, therefore our algorithm is very efficient.

L_{max}	#	$T_{50\%}$ (ns)	T (ns)
7	3	1.3381	0.3592
15	5	1.2815	0.1635
31	4	1.3217	0.1635
63	6	1.3018	0.0967
127	8	1.2970	0.0670

Table 2. The optimal buffer insertion to minimize transition time.

In Table 2, we summarize the result where the optimal buffer insertion is chosen such that the final transition time is minimized. It is interesting to note that if we can tolerate a little bit longer delays, we can obtain results which have much faster transition times than those shown in Table 1, with the cost of adding few more buffers. The buffer insertion scheme for $L_{max} = 127$ is shown in Figure 6(b).

6. Conclusion

We have presented a graph based algorithm for optimal buffer insertion. The optimal buffer insertion problem can be reduced to a shortest path problem. In our algorithm, signals are accurately represented by a finite ramp which has two parameters, shift time and transition times. Any accurate delay model including SPICE simulations can be used in our algorithm. Our algorithm can be easily extended to deal with simultaneous buffer insertion and wire-sizing. Our algorithm is very efficient, and the optimal solution can be obtained within a few seconds.

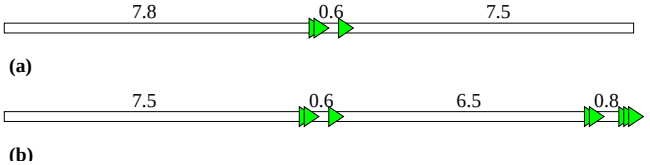

Figure 6. The optimal buffer insertion results. (a). Minimal delay. (b). Minimal transition time. All lengths are in mm.

References

[1] C.J. Alpert, A. Devgan and S.T. Quay, Buffer Insertion with Accurate Gate and Interconnect Delay Computation, *Proc. ACM/IEEE Design Automation Conf.*, pp.479-484, 1999.

[2] C.J. Alpert, A. Devgan and S.T. Quay, Buffer Insertion for Noise and Delay Optimization, *Proc. ACM/IEEE Design Automation Conf.*, pp.362-367, 1997.

[3] C.-P. Chen and D.F. Wong, Optimal Wire-sizing Function with Fringing Capacitance Consideration, *Proc. ACM/IEEE Design Automation Conf.*, pp.604-607, 1997.

[4] C.-P. Chen and N. Menezes, Noise-Aware Repeater Insertion and Wire Sizing for On-Chip Interconnect Using Hierarchical Moment-Matching, *Proc. ACM/IEEE Design Automation Conf.*, pp.502-506, 1999.

[5] C.C.N. Chu and D.F. Wong, Closed Form Solution to Simultaneously Buffer Insertion/Sizing and Wire Sizing, *Int. Symp. on Physical Design*, pp.192-197, 1997.

[6] C.C.N. Chu and D.F. Wong, A New Approach to Simultaneous Buffer Insertion and Wire Sizing, *Proc. IEEE Int. Conf. on Computer Aided Design*, pp.614-621, 1997.

[7] W.C. Elmore, The Transient Response of Damped Linear Network with Particular Regard to Wide-band Amplifier, *Journal of Applied Physics*, vol.19, pp.55-63, 1948.

[8] Y. Gao and D.F. Wong, Wire-Sizing Optimization with Inductance Consideration Using Transmission Line Model, *IEEE Trans. on Computer-Aided Design*, 1999.

[9] Y. Gao and D.F. Wong, A Fast and Accurate Delay Estimation Method for Buffered Interconnects, *Proc. IEEE Asia and South-Pacific Design Automation Conf.*, 2001.

[10] A.B. Kahng and S. Muddu, Delay Models for MCM Interconnects When Response is Non-monotone, *IEEE Multi-Chip Module Conf.*, pp.102-107, 1997.

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

[12] N. Menezes and C.-P. Chen, Spec-Based Repeater Insertion and Wire Sizing for On-Chip Interconnect, *Proc. Intl. Conf. on VLSI Design*, pp.476-483, 1999.

[13] P.R.O'Brien and T.L. Savarino, Modeling the Driving Point Characteristic of Resistive Interconnect for Accurate Delay Estimation, *Proc. IEEE Intl. Conf. on Computer Aided Design*, pp.512-515, 1989.

[14] J. Qian, S. Pullela and L. Pillage, Modeling the "Effective Capacitance" for the RC Interconnect of CMOS Gates, *IEEE Trans. on Computer-Aided Design*, Vol.13, No.12, pp.1526-1535, 1994.

[15] L.P.P.P. van Ginneken, Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay, *Intl. Symp. Circuits and Systems*, pp.865-868, 1990.

[16] N.H.E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design* , 2nd Edition, Addison-Wesley Publishing Company, 1993.

[17] Q. Yu and E.S. Kuh, Exact Moment Matching Model of Transmission Lines and Application to Interconnect Delay Estimation, *IEEE Trans. on VLSI*, Vol.3, No.2, pp.311-322, 1995.