Accurate Estimation of Global Buffer Delay within a Floorplan

Charles J. Alpert[1], Jiang Hu[2], Sachin S. Sapatnekar[3], and C. N. Sze[2]
[1]IBM Corp., 11501 Burnet Road, Austin, TX 78758, alpert@us.ibm.com
[2]Texas A&M University, EE Department, College Station, TX 77843, {jianghu, cnsze}@ee.tamu.edu
[3]University of Minnesota, ECE Department, 200 Union St., SE Minneapolis, MN 55455, sachin@ece.umn.edu

Abstract

Closed formed expressions for buffered interconnect delay approximation have been around for some time. However, previous approaches assume that buffers are free to be placed anywhere. In practice, designs frequently have large blocks that make the ideal buffer insertion solution unrealizable. The theory of [12] is extended to show how one can model the blocks into a simple delay estimation technique that applies both to two-pin and to multi-pin nets. Even though the formula uses one buffer type, it shows remarkable accuracy in predicting delay when compared to an optimal realizable buffer insertion solution. Potential applications include wire planning, timing analysis during floorplanning or global routing. Our experiments show that our approach accurately predicts delay when compared to constructing an realizable buffer insertion with multiple buffer types.

1. Introduction

Buffer insertion is becoming an ever critical component of physical synthesis for timing closure and design planning (see Cong et al. [4] for a survey). Saxena et al. [13] estimate that the distance between buffers continues to shrink rapidly. One must be able to efficiently and accurately assess the impact of buffer insertion on a design, whether in terms of floorplanning, resource allocation, timing estimation or within an actual buffer insertion heuristic.

To this end, several works (e.g., [1][3][6][8][12]) have explored closed form expressions for buffer insertion on a line. None of these works model blockages in the layout. Given the advent of SoC chip design and the trends towards large memory arrays, IP cores, and hierarchical design, an ever increasing percentage of the layout is covered by blocks in which buffers cannot be inserted (though routes may cross over). A large blockage can cripple a route's ability to meet timing since delay is quadratic in length when no buffers are inserted, but linear in length for optimal buffer insertion. Blockages are now a first order delay effect and must be taken into account for any buffer estimation technique to be sufficiently accurate. Several works, e.g., [7][10][11][14], have explored the problem of buffer insertion when there are constraints on the buffer positions, but none of them address the problem of delay estimation.

Our work begins from Otten's theoretical result [12] that in an optimal buffering, delay is linear in terms of length. With large blockages and multi-fanout nets, a linear delay solution is not necessarily realizable. It is not at all immediately obvious how to overcome these limitations, and whether such an extension (even if it were possible) would even be valid in a real design methodology. The primary contribution of this work is to extend Otten's theory to predict interconnect delays for multi-fanout nets in the presence of blockages, and to *validate it on real industrial test-cases*. The end result of this work is a fast and simple formula that is proven on real design scenarios, and can be of practical use in early design planning.

*This work is partially supported by the SRC under contract 2003-TJ-1124.

The following key assumptions actually impose little error when compared to an actual buffer insertion solution:

1. **Smaller blocks ignored**: Let L_{opt} be the spacing between consecutive buffers that obtains optimal signal propagation speed. Blocks with width less than L_{opt} can safely be ignored. While this may cause buffers to be inserted at distance less than the optimal spacing, having multiple buffers in the library allows the optimal linear delay to still be achievable. An example is shown in Figure 1. As long as the blocks are smaller than L_{opt}, the optimal realizable buffering in (b) will have delay very close to the ideal buffering of (a). Hence we can assume the delay model of (a) to approximate (b).
2. **Single buffer type**: A single buffer type that yields the fastest point to point delay is sufficient for modeling. It turns out that the more buffer types that are actually in the library, the better the single buffer type approximation. As shown in Figure 1(b) different size buffers may be required to buffer distances that are less than L_{opt}. Having additional buffers in the library allows the situation in (b) better recover from its perturbation from the ideal buffering in (a).
3. **Block locations ignored**: whether blocks are closer to the source or sink has little effect on the actual buffered delay, so we ignore this effect.
4. **Infinitesimal decoupling buffers**: Buffers for decoupling capacitance off the critical path can be modeled to have zero input capacitance. This may lead to slight underestimation of delay, but the effect is almost negligible.
5. **Larger block front-to-back buffering**: A block with width larger than L_{opt} will cause a linear delay model to break down. To optimize the delay across such a block, and optimal buffering will almost assuredly place a buffer right before and right after the block. Hence the delay across the large block can be modeled separately from the rest of the Steiner route.

Despite the inaccuracy it would seem these assumptions impose, our simple linear time estimation technique shows high accuracy when compared to realizing a buffer insertion solution with van Ginneken's algorithm [15].

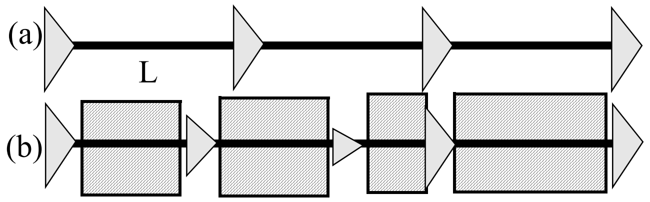


Figure 1 Example of (a) an optimally buffered line with equal spaced buffers and (b) an optimal realizable buffering when blockages are present. Note that unequal buffer sizes may be used here.

This approach has several potential applications. For example, it can assess the timing cost of different block configurations during floorplanning or assess different Steiner routes during wire planning. A global router can use this to decide which of several possible Steiner tree constructions is likely to yield the best timing result. One may want to do timing analysis of a floorplan and/or placement without having to actually perform the buffer insertion for a net. One could embed the formula into a placement algorithm. Finally, the recent work of applies this approach in a Steiner tree construction that navigates the environment as a precursor to buffer insertion.

2. Closed Form Formula for Two-Pin Nets

Consider buffering a line of length L. Assume a given "ideal" buffer [12] (or inverter) b that is optimal for signal propagation speed on a wire. Let the intrinsic resistance and input capacitance of b be given by R_b and C_b, respectively. Assume the intrinsic buffer delay is zero since this is a first order approximation. One could also add an intrinsic resistance term and derive alternative formulae.

The unit wire resistance and capacitance are given by R and C, respectively. We make the following assumptions:

- The driver of the net has the same driver resistance as b. If this assumption is incorrect, that may indicate a design flaw. Too large a resistance means the driver should probably be powered up until the resistance is close to R_b. Even if the gate cannot be properly sized, it will likely need a few buffers (in which the last buffer is of size b) as close as possible to the driver to power up the signal in order to drive the line. Similarly, if the resistance is much lower than b, this indicates that the gate is likely overpowered and can be powered down.
- The sink of the net has input capacitance C_b. A different value should not significantly change the value, especially for a long line. Significantly different input capacitances also may potentially indicate an ill-sized sink. Overall, we find that sink capacitance gets overshadowed by wire capacitance on nets that require buffering.
- The intrinsic buffer delay is zero. This terms tends to be dominated by the R_bC_b term. However, an intrinsic delay term can easily be incorporated if desired.

2.1 Delay Formula with No Blockages

The following result was also derived by Otten [12]. We present it here for completeness.

Theorem 1: The delay $D(L)$ function of an optimally buffered line of length L with no blockages asymptotically approaches the linear function of the design and buffer parasitics given by:

$$D(L) = L(R_bC + RC_b + \sqrt{2R_bC_bRC}) \tag{1}$$

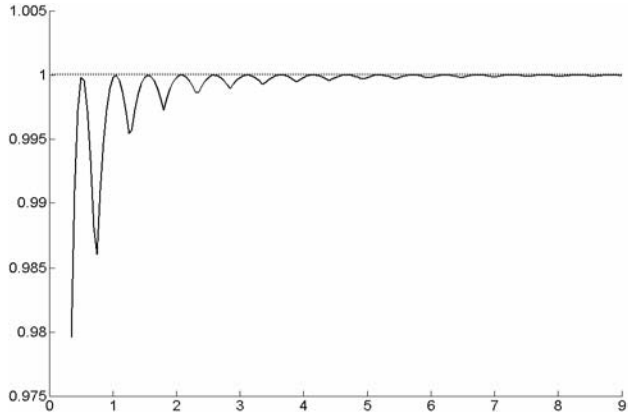
Proof: Let k be number of stages (so there are $k-1$ buffers) that results in the optimal delay along L, as in Figure 1(a). Several works have proven that the optimal buffer configuration spaces the buffers at equal distances. Since the source and sink have the same parasitics as b, all stages have the same delay. The length of wire between consecutive stages is L/k. The delay on the line is given by k times the sum of the buffer delay and the wire delay.

$$D(L) = k\left(R_b\left(\frac{CL}{k} + C_b\right) + \frac{RL}{k}\left(\frac{CL}{2k} + C_b\right)\right) \tag{2}$$

We wish to find the optimal number of buffers k. Taking the derivative with respect to k, setting the expression to zero, and solving for k yields the optimal number of buffers

$$k = L\sqrt{\frac{RC}{2R_bC_b}} \tag{3}$$

Obviously, if k is not an integer one may need to try rounding k up or down and see which yields the best delay. Substituting Equation (3) into Equation (2) yields the theorem.



The length of a two-pin net in millimeters.

Figure 2 Ratio of Equation (1) to (2) as a function of wirelength.

Observe that $D(L)$ is a lower bound on the realizable delay. A nice property of Theorem 1 is that it is independent of the number of buffers. Of course, this will introduce some error because the delay in (1) is not realizable when the optimal number of stages is not an integer. However, the error is actually quite small, as illustrated by Figure 2.

The figure presents the ratio of the delay according to the estimation formula in Equation (1) to the delay of Equation (2). As L goes to infinity, the ratio goes to one, meaning the error goes to zero. Note that the maximum error occurs when the optimum number of buffers k is not an integer. However, the realizable delay will actually be even less than in Equation (2) if one permits additional buffer sizing. For example, if the number of buffers $k = 3.5$, then one may size up the buffer type until the ideal number is three or size down until it is four, thereby achieving a slightly better delay. Even without sizing though, Equation (1) is within 0.5% of Equation (2) when more than one buffer is required.

This result is perhaps not surprising, given the observations of Cong et al. [5]. They show that a fairly large "feasible region" exists for each buffer to be manipulated without suffering significant degradation in timing. Our example bears this out as buffers are shifted slightly when k is not an integer in order to get equal spacing between buffers. This shifting results in close to the ideal delay of Equation (1).

Corollary 1: The optimum spacing L_{opt} between buffers is

$$L_{opt} = \sqrt{\frac{2R_bC_b}{RC}}. \tag{4}$$

2.2 Delay Formula with Blockages

Next we consider inserting a block of width w somewhere on the line L. This notion can be generalized to include jogs and bends as in Figure 3. Let $l(u, v)$ be the length on the route from u to v. In the figure, we consider L to be $l(s_0, s_1)$ and w to be $l(x_0, x_1)$. We wish to derive a delay formula that is a function solely of L and w. Our strategy is as follows:

- For $w < L_{opt}$, we assume that buffers can be placed (and potentially sized) in such a way as to avoid the blockage while only suffering a nominal delay penalty. Hence, we ignore w and just use Theorem 1.
- For $w \geq L_{opt}$, we try placing a buffer immediately before and after a blockage. This minimizes the quadratic effect on delay that the width of the blockage has. For the rest of the line, we simply invoke Theorem 1, again accepting the potential error from the inability to have the optimal number of buffers be a non-integer.

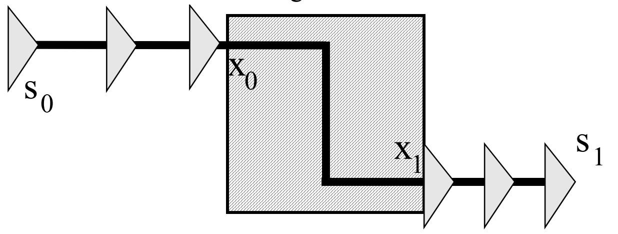


Figure 3 Buffering scheme on a route of length $L = l(s_0, s_1)$ with a single blockage spanning length $w = l(x_0, x_1)$.

For the second scenario ($w \geq L_{opt}$), the buffered delay is given by the buffered delays of the unblocked wires plus the delay needed to cross the blockage. The latter term is given by the Elmore delay:

$$ED(w) = R_b(Cw + C_b) + Rw\left(\frac{Cw}{2} + C_b\right) \tag{5}$$

Given a set of blockages W with crossing width $w \geq L_{opt}$, then one can assume the existence of a single buffer before and after each blockage and summing the pieces together. We overload the D function so that $D(L)$ is the formula in Equation (1) for no blockages and $D(L, W)$ is the following delay for a set of blockages W.

Blockage Buffered Delay Formula:

$$D(L, W) = D(L - L_W) + \sum_{w \in W} ED(w) \quad \text{where} \quad L_W = \sum_{w \in W} w \tag{6}$$

Unlike Theorem 1, this formula is not a lower bound on the actual achievable delay. It could conceivably over-estimate delay since inserting buffers right after one blockage and right before another may result in overly tight buffer spacing. However, the delay from optimal buffered solution is rarely smaller than that from Equation (6) when $w \geq L_{opt}$, as we demonstrate in the next section.

3. Two-Pin Experiments

We use the following parameters from 100 nanometer technology [6]: $R = 0.184\ \Omega/\mu m$, $C = 0.0715\ ff/\mu m$, $R_b = 246.3\ \Omega$, and $C_b = 7.2\ ff$. For these values, Equation (4) yields $L_{opt} = 519\mu m$. Assume that the sink and source are both buffers b. We first illustrate the accuracy of Equation (6) is accurate, even though it is independent of the blockage location. For each possible blockage location, we compute $D(L, W)$ and the optimal delay according to van Ginneken's algorithm (using only buffer type b).

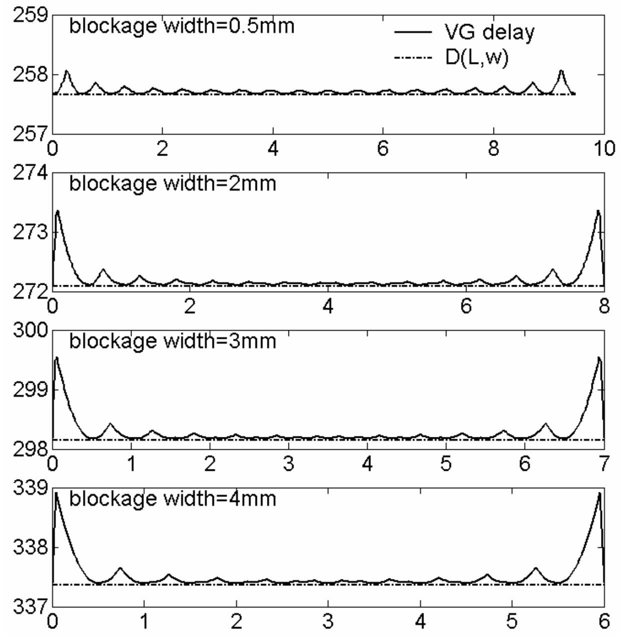


Figure 4 Comparison of the blockage buffered delay formula with van Ginneken's algorithm for the case of a single blockage on a 2-pin net with length ten mm.

Figure 4 shows this for a ten mm wire for blockages with widths 0.5, 2.0, 3.0, and 4.0 mm. The horizontal axis give the location of the blockage in terms of the distance from the source. We observe the following. First, for all examples, the closed form of Equation (6) is a tight lower bound. Next, the error is less than 1% for all blockage widths and blockage placement. Finally, the maximum error occurs at the tail ends because this is where one may have to either insert two buffers that are too close together or drive a distance that is actually longer than the blockage. With a library of multiple buffer types for van Ginneken's algorithm, this small error is reduced even further. Thus, for a single blockage the error is insignificant.

Next, consider the scenario when multiple blockages cover a significant part of the wire. We generated ten different instances with either 3 or 4 blockages, covering a large percentage of a 12 mm wire. The ten cases are shown in Table 1, where the second column gives the blockage widths and the third column gives the corresponding distances from the source). For each case, we report the Blockage Buffered Delay formula and the optimal delay according to van Ginneken's algorithm in columns four and five. From the last error column we see that our formula is well within one percent of optimal for all ten cases.

test case	Block widths (mm)	Positions (mm)	Eq. (6) (ps)	van Gin delay (ps)	Error (%)
1	1.8/4.0/2.9	0.1/2.2/6.7	437.0	438.5	0.35
2	2.5/4.0/2.9	0.3/3.2/8.7	451.9	452.5	0.11
3	0.5/4.7/2.1	1.3/2.2/9.7	440.6	441.5	0.21
4	3.5/4.7/2.0	0.0/4.2/9.7	497.0	497.8	0.14
5	4.5/0.7/3.0	0.5/6.2/8.7	454.1	454.7	0.12
6	2.5/2.1/2.9/1.1	0.3/3.2/6.7/10.0	390.9	391.6	0.16
7	2.5/1.1/5.9/0.5	0.0/3.2/4.7/11.0	527.7	528.1	0.08
8	2.6/4.4/0.9/1.8	0.3/3.2/8.7/10.2	448.5	449.2	0.15
9	1.5/3.3/0.9/4.2	0.3/2.2/5.7/7.3	456.5	457.8	0.28
10	1.5/3.3/3.9/2.2	0.0/2.2/5.7/9.8	456.5	461.7	0.11

Table 1 Comparison of the blockage buffered delay formula with van Ginneken's algorithm for multiple blockages on a 2-pin net with length 12 mm.

We have effectively shown that for single and multiple blockages, the error from our formula is insignificant.

4. Linear Time Estimation for Trees

We now show how to extend the two-pin formulae to trees in the presence of blockages. Two convenient properties of the following estimation technique are that:

- It can be decomposed into a summation of piecewise components, just like the Elmore delay, thereby enabling efficient optimization algorithms.
- The delay can be broken into the sum of the delays on a given path, allowing one to compute the worst slack of the tree in a single bottom-up traversal.

Let $T(V, E)$ be a Steiner tree with n nodes and source node s_0 and sinks $s_1, \ldots, s_k$. Let $RAT(s_i)$ be the required arrival time for sink s_i and let $p(v)$ be the parent of node $v \in V - \{s_0\}$. The quality of a given buffer solution is typically measured by the slack at the source node, which is given by

$$q(s_0) = min_{1 \le i \le k}\{RAT(s_i) - Delay(s_0, s_i)\} \quad (7)$$

where $Delay(s_0, s_i)$ is the buffered delay from the s_0 to s_i.

The key idea is to assume that if a path from s_0 to s_i is the most critical, that all sub-trees off the critical path will be decoupled. To achieve the lowest delay to the critical sink, the decoupling buffer should have the minimum possible input capacitance; we assume input capacitance zero. We show in Section 5, that this is a second order effects, as compared to the first order blockage effect.

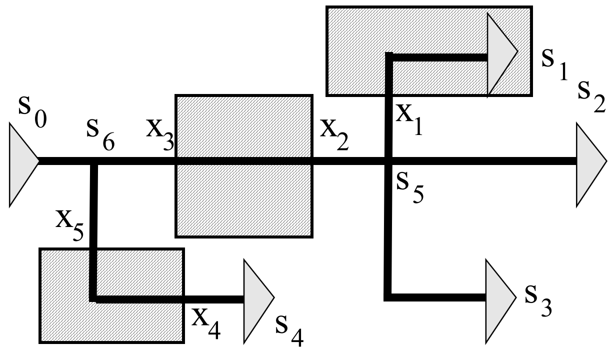


Figure 5 Multi-sink tree with only unblocked Steiner points.

4.1 Case 1: Unblocked Steiner Points

Consider the case where all Steiner points are unblocked, as in the four-sink example of Figure 5. Here, all decoupling of branches can be accomplished by placing the buffer right near the Steiner point. Hence, the delay to a given sink can be broken piecewise into the sum of its sub-paths. For example, the delays in Figure 5 are given by:

$$\begin{aligned} Delay(s_0, s_1) &= D(l(s_0, s_1), \{l(x_3, x_2), l(x_1, s_1)\}) \\ Delay(s_0, s_2) &= D(l(s_0, s_2), \{l(x_3, x_2)\}) \\ Delay(s_0, s_3) &= D(l(s_0, s_3), \{l(x_3, x_2)\}) \\ Delay(s_0, s_4) &= D(l(s_0, s_4), \{l(x_5, x_4)\}) \quad (8) \end{aligned}$$

where $D(L, W)$ is given by Equation (6).

4.2 Case 2: Blocked Steiner Points

Now consider when Steiner points may lie inside blockages, as in Figure 6. In this case, decoupling may only occur outside of the blockage after incurring potentially significant wirelength. This is modeled by keeping track of the off-path capacitance and multiplying it by the upstream resistance inside the blockage. Also, we need to add the delay from the extra capacitance loading on the (imaginary) driving buffer. Define the function $OD(l_r, l_c)$ to be the delay from the off-path capacitance inside a blockage as:

$$OD(l_r, l_c) = (R \cdot l_r + R_b)(C \cdot l_c) \quad (9)$$

For example, some of the delays in Figure 6 are given by:

$$
\begin{aligned}
Delay(s_0, s_1) &= D(l(s_0, s_1), \{l(x_6, x_5), l(x_1, s_1)\}) \\
&\quad + OD(l(x_6, s_5), l(s_5, x_2) + l(s_5, x_3)) \\
&\quad + OD(l(x_6, s_6), l(s_6, x_4)) \\
Delay(s_0, s_4) &= D(l(s_0, s_4), \{l(x_6, x_4)\}) \\
&\quad + OD(l(x_6, s_6), l(s_6, x_5) + l(s_5, x_2) + l(s_5, x_3)) \qquad (10)
\end{aligned}
$$

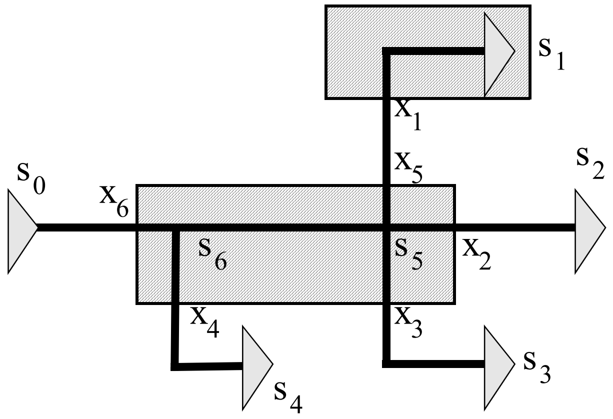


Figure 6 Multi-sink tree example with blocked Steiner points.

4.3 Linear Time Estimation Algorithm

The examples of Figure 5 and Figure 6 show that the estimation can be expressed as a formula to find the delay to any sink. It is not as clear how to compute the slack at the source without having to compute the delay to each individual sink. We now present a linear time algorithm to compute the slack at the source in a single bottom-up tree traversal. The key component is to recognize that at any given Steiner point, the most critical downstream path can be determined because any upstream delay will be the same for all sinks downstream from the Steiner point.

For each node v, let $q(v)$ denote the slack at node v, and let $C(v)$ denote the sub-tree capacitance downstream from v that is in the same blockage as v. In Figure 6,

$$
\begin{aligned}
C(s_5) &= C(l(s_5, x_3) + l(s_5, x_5) + l(s_5, x_2)) \text{ and} \\
C(s_6) &= C(s_5) + C \cdot (l(s_6, x_4) + l(s_6, s_5)) \qquad (11)
\end{aligned}
$$

Note that the input buffer capacitance C_b is not stored in $C(v)$, but is only invoked when making a delay calculation. We only want to consider this additional capacitance on the critical path, but not for the whole sub-tree, since the non-critical paths can be decoupled with much smaller buffers.

We assume the edges in the tree are segmented such that whenever a blockage in W intersects a tree edge, the edge is broken into two edges incident to an intermediate boundary node (as the x_i's are in Figure 5 and Figure 6). So each edge (u, v) lies either completely inside or outside a blockage in W. Boundary nodes lie outside W. A node lies inside W only if it is completely inside a block in W, e.g., in Figure 6 only s_5 and s_6 lie inside W.

```
Inputs: $T(V, E)$ Given Steiner tree
        $R$, $C$ resistance/ capacitance per unit length
        $R_b$, $C_b$ resistance/ input capacitance of buffer $b$
        $RAT(v)$ for each sink $v$
        $W$ set of blockages

Variables: $q(v)$ slack at node $v$
           $q_i(v)$ slack at node $v$ on path to child $u_i$
           $C(v)$ in-blockage capacitance downstream from $v$

Let: $\alpha = R_b C + R C_b + \sqrt{2 R_b C_b R C}$

1. for each $v \in V$ (in bottom-up order) do
     set $C(v) = 0$
2.   if $v$ is a sink,
       set $q(v) = RAT(v)$.
3.   if $v$ has children $u_1, \ldots, u_k$ for $k \ge 1$, then
       for $i = 1$ to $k$ do
4.       if $(u_i, v) \notin W$ then
           $q_i(v) = q(u_i) - \alpha \cdot l(v, u_i)$
         if $(u_i, v) \in W$, then
           $ED = R \cdot l(u_i, v)(C \cdot l(v, u_i)/2 + C(u_i) + C_b)$
           set $q_i(v) = q(u_i) - ED$
           set $C(v) = C(v) + C \cdot l(v, u_i) + C(u_i)$
5.     let $j$ be such that $q_j(v) = min_{1 \le i \le k}\{q_i(v)\}$
       set $q(v) = q_j(v)$
       if $v \notin W$ and there exists $u_j$ s.t. $(v, u_j) \in W$ then
         set $q(v) = q(v) - R_b(C(v) + C_b)$
         set $C(v) = 0$
6.   if $v$ is the source, then return $q(v)$
```

Figure 7 Linear time estimation algorithm for trees.

The algorithm is shown in Figure 7. Instead of using the formulae from Section 2, the delay is computed piecewise since this affords the simplest direct implementation. Step 1 visits each node v in a bottom-up tree traversal, initializing the downstream capacitance $C(v)$ to zero. Step 2 handles the case where v is a sink, initializing the slack to the required arrival time. Step 3 handles multiple children and iterates through the children $u_1, \ldots, u_k$ of v. Step 4 updates upstream information when going from node u_i to v via the intermediate variable $q_i(v)$. If the edge (u_i, v) is not in a blockage, the downstream in-blockage length is zero and the slack is updated by the linear delay from v to u_i. The slack is updated to include the Elmore delay from v to the critical node downstream from u_i that is just outside the blockage containing v. Finally, for edges (u_i, v) that lie within blockages, all downstream capacitance is summed in $C(v)$.

Step 5 then identifies the child u_j of v that is the ancestor of the most critical sink, and the slack at v is then set. Finally, if v is not in a blockage, but the edge (u_j, v) is, then one must incorporate into the slack the additional delay required for a buffer just outside the blockage to drive. Since v is not outside the blockage, its downstream capacitance is then set to zero. Finally, Step 6 returns the slack at the source. The time complexity is linear in the number of nodes.

5. Experiments for Multi-Sink Nets

We call our estimation algorithm in Figure 7 BELT for Blockage Estimation in Linear Time. We consider three other buffered slack calculations.

1. One can compute the estimation formula while ignoring blockages. This in effect reduces to the estimations of [5][12], whereby one just looks at the length of each path and performs an optimal buffering as if it were a 2-pin net. Since this is essentially the BELT estimation without the blockages, we call this formula ELT.
2. As in Section 3, we run van Ginneken's algorithm using the single buffer type b, and call this VG1 since there is one buffer type.
3. In practice, we have the ability to run actual buffer insertion with additional buffers types. We generated three additional smaller buffers (since b is already a larger buffer) to use with van Ginneken framework. We call this algorithm VG4.

All codes were written in C++, and compiled using g++ version 2.95 on a Sun Ultra-4 running SunOS 5.7.

For the following experiments, the required arrival times were chosen to be the same for each sink since this actually increases the likelihood of error in the delay estimation formula. If one sink is substantially more critical, then this sink will have all off-path branches decoupled, making it an easier problem. Consequently, instead of reporting slack, we report the maximum path delay (which also makes interpreting results more intuitive).

5.1 Results on Random Nets

Our first experiment examines randomly generated nets. First we created a simple artificial floorplan. The plan has 16 high level square blocks, each five millimeters on a side. The block are arranged in a regular pattern on a square layout that is 21 millimeters on a side. Thus, there is sufficient space in the alleys ($0.25mm$ wide) between blocks to allow buffer insertion. This type of layout loosely corresponds to the kind of behavior one might expect from a large chunky hierarchical design.

Next, we generated nine nets each of size three through ten pins. We ran the four different algorithms, ELT, BELT, VG1

net sinks	WL (μm)	% Blk	ELT	%ELT/ BELT	BELT	%BELT/ VG4	BELT ln2	%BELT_ln2/ VG4_SPICE	VG1	VG4	VG4 SPICE
3	29104	88.5	552.8	45.0	1225.9	99.4	849.7	93.7	1235.0	1232.8	906.9
4	41983	93.3	617.0	43.5	1416.6	99.6	981.9	92.6	1428.7	1422.1	1060.6
5	39904	90.2	512.8	42.1	1216.6	99.2	843.3	93.2	1230.2	1225.9	904.5
6	46559	90.2	569.8	43.9	1295.6	99.4	898.1	93.0	1308.0	1303.7	965.5
7	50373	88.9	548.6	42.2	1299.3	99.2	900.6	93.5	1314.2	1309.4	963.0
8	59190	91.1	663.0	43.0	1541.7	98.9	1068.6	95.7	1558.0	1551.2	1117.0
9	54659	90.5	539.0	39.8	1353.0	99.4	937.8	94.0	1375.3	1368.2	997.7
10	65350	94.0	595.1	41.7	1426.4	98.9	988.7	95.1	1449.1	1441.8	1040.2

Table 2 Experiments on randomly generated nets. Each row represents the average of nine different nets.

net name	WL (μm)	Sinks	% Blk	ELT	BELT	%BELT/ VG4	BELT ln2	%BELT_ln2/ VG4_SPICE	VG1	VG4	VG4 SPICE
mcu0	50540	18	89.9	380	822	95.1	569.9	88.6	872	864	643.4
mcu1	41780	19	96.2	492	1052	97.4	729.3	92.6	1084	1080	787.3
n107	14870	17	97.6	257	361	91.4	250.2	85.8	396	395	291.7
n189	64700	29	83.8	573	1486	96.9	1029.8	91.9	1556	1532	1120.9
n313	69430	19	96.6	587	1821	99.0	1262.3	95.6	1850	1840	1319.7
n786	53110	32	96.4	1126	3574	92.3	2477.2	85.1	3880	3873	2911.3
n869	42180	21	96.6	1042	2605	92.6	1805.3	85.1	2816	2813	2122.5
n870	45230	21	97.3	972	2326	93.1	1612.4	85.4	2498	2498	1887.3
n873	49290	43	78.0	527	1363	99.1	944.6	92.1	1381	1375	1026.1
poi3	63600	20	96.8	1256	3746	97.4	2596.6	89.6	3854	3847	2898.1
big1	195300	88	85.8	1143	5920	97.6	4103.6	99.8	6115	6063	4111.0
big2	122500	79	93.1	545	1577	96.0	1093.1	95.9	1657	1643	1139.9
big3	95320	63	94.1	403	1415	96.8	980.5	96.7	1478	1460	1014.0

Table 3 Experiments for 13 nets from an industry design.

and VG4, on each net and summarize the results in Table 2. In each case we set the driver strength and the sink size to be equal to buffer b. The Steiner topology was generated using the C-Tree algorithm which ignores blockages [2]. Delay calculations are for 0.10 micron technology [6].

The table presents a single row summarizing the average of nine different nets each having the specified number of sinks. For each net, we report the average wirelength and percent of the net that was blocked. For the four algorithms we present the average maximum delay for the net. The solutions of VG4 are also evaluated by SPICE model and the results are shown in the rightmost column. The SPICE results are based on 50% Vdd signal delay. Column 5 gives the ratio of ELT to BELT delay as a percentage. Note that by definition ELT will always be less than BELT. The percentage ratio of BELT to VG4 delay is listed in column 7 for each case. It is well known that the Elmore delay is an upper bound of real delay and people often multiply ln2 with the Elmore delay to reflect 50% Vdd signal delay. When comparing the timing performance of two Steiner trees for a same net, the scaling of ln2 does not affect the conclusion of the comparison. In order to have more fair comparison with the SPICE results, we report the ratio of the BELT results scaled by ln2 to the SPICE based VG4 results in column 9. We observe the following:

- By comparing the ratio of ELT to BELT in column 5, observe if one ignores blockage, the errors are typically off by over a factor of two. Of course the degree of the error will depend on the size of the blocks. Clearly, ignoring blocks causes gross underestimation of the achievable delay.
- Comparing BELT to VG1, we see that the delay estimation is quite accurate, and tends to underestimate the achievable delay by 1.1% on average.
- Comparing BELT to VG4, we see that the error is reduced even further, to 0.8% on average.
- When compared to SPICE based VG4, the error of BELT is always less than 8%.

Clearly, the accuracy of BELT is sufficient while the accuracy of an estimation technique that is not blockage aware begins to suffer fairly significant underestimation. This effect becomes magnified when the blockage map has large blocks that may correspond to IP cores or memory.

5.2 Results on Large Real Nets

Our next experiments use the Steiner trees for a set of the industrial nets reported in [2] and [11]. We perform the same set of experiments as in Section 5.1 and report the results in Table 3. This time the nets are listed on an individual basis.

We observe the following:

- On average, the ELT/BELT percentage is 36.2%, which means that blockage has on average about a 64% impact for these nets.
- For some cases, the impact of blockage is not that significant, e.g., for net n107 ELT is a reasonable estimate. For others, it is quite large, e.g., netbig1 has an ELT/BELT percentage of 19.3%. In this case, we see that ELT underestimates the (realizable) VG4 delay by 81%, while BELT underestimates the VG4 delay by 3%.
- On average the error of BELT compared to VG4 is 5.2%, while on average the VG4 delay is almost a factor of three higher than that predicted by ELT.
- Compared to SPICE based VG4, the error of BELT is 9% on average.

These experiments illustrates that our estimation technique is sufficiently accurate for design planning, while ignoring blockages is prohibitively costly.

Finally, note how efficient the estimation technique is. The total runtime in seconds for running the above 13 test cases was 0.24, 23.0 and 29.0 for BELT, VG1 and VG4, respectively. In other words, BELT is about 100 times faster than running an actual buffer insertion algorithm. For medium sized nets, such as n786 and n869, the runtime of VG4 plus SPICE simulation is over 10000 times slower than the runtime of BELT.

6. Conclusions

We presented closed form formulae for estimating the achievable buffered delay when buffering restrictions exist in the layout. We demonstrate that adding blockages to the layout can cause significant error in estimation techniques that ignore the blockage terrain. We also showed that our technique is a lower bound, has an error of less than one percent for two-pin nets, and has only a few percent error for multi-sink nets.

References

[1] C. J. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion", *ACM/IEEE DAC* pp. 588-593, 1997.

[2] C.J. Alpert, G. Gandham, M. Hrkic, J. Hu, A.B. Kahng, J. Lillis, B. Liu, S.T. Quay, S.S. Sapatnekar and A.J. Sullivan, "Buffered Steiner trees for difficult instances", *IEEE Transactions on CAD*, 21(1):3-14, January 2002.

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

[4] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance Optimization of VLSI Interconnect Layout", *Integration: the VLSI Journal*, 21, 1996, pp. 1-94.

[5] J. Cong, T. Kong, and D. Z. Pan, "Buffer block planning for interconnect-driven floorplanning", *IEEE/ACM Conf. on Computer-Aided Design*, pp. 358-363, 1999.

[6] J. Cong and D. Z. Pan, "Interconnect performance estimation models for design planning" *IEEE Transactions on CAD*, 20(6):739-752, June 2001.

[7] J. Cong and X. Yuan, "Routing tree construction under fixed buffer locations", *ACM/IEEE DAC*, pp. 379-384, 2000.

[8] S. Dhar and M. A. Franklin, "Optimum buffer circuits for driving long uniform lines", *IEEE Journal of Solid State Circuits*,

26(1):33-38, January 1991.

[9] W. C. Elmore, “The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifiers”, *J. Applied Physics*, 19, 1948, pp. 55-63.

[10] M. Hrkic and J. Lillis, “Buffer tree synthesis with consideration of temporal locality, sink polarity requirements, solution cost and blockages”, *ACM/IEEE ISPD*, pp. 98-103, 2002.

[11] J. Hu, C.J. Alpert, S.T. Quay and G. Gandham, “Buffer insertion with adaptive blockage avoidance”, *ACM/IEEE International Symposium on Physical Design*, pp. 92-97, 2002.

[12] R. H. J. M. Otten, “Global Wires Harmful?”, *ACM/IEEE Intl. Symposium on Physical Design*, pp. 104-109, 1998.

[13] P. Saxena, N. Menezes, P. Cocchini and D. A. Kirkpatrick, “The scaling challenge: can correct-by-construction design help?”, *ACM/IEEE ISPD*, pp. 51-58, 2003.

[14] X. Tang, R. Tian, H. Xiang and D.F. Wong, “A new algorithm for routing tree construction with buffer insertion and wire sizing under obstacle constraints”, *IEEE/ACM International Conf. on Computer-Aided Design*, pp. 49-56, 2001.

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

[16] C. J. Alpert, M. Hrkic, J. Hu, and S. T. Quay, “Fast and Flexible Buffer Trees that Navigate the Layout Environment”, *IEEE/ACM DAC,* pp. 24-29, 2004.