Is Wire Tapering Worthwhile?

Charles J. Alpert
IBM Austin Research Laboratory
Austin, TX 78717
alpert@austin.ibm.com

Anirudh Devgan
IBM Server Group
Austin, TX 78717
devgan@austin.ibm.com

Stephen T. Quay
IBM Server Group
Austin, TX 78717
quayst@austin.ibm.com

Abstract

Wire sizing and buffer insertion/sizing are critical optimizations in deep submicron design. The past years have seen several studies of buffer insertion, wire sizing, and their simultaneous optimization. When wiring long interconnect, tapering, i.e., reducing the wire width as the distance from the driver increases, has proven effective. However, tapering is not widely utilized in industry since it is difficult to integrate into a complete routing methodology. This work examines the benefits of wire sizing with tapering when combined with buffer insertion. We perform several experiments with actual IBM technologies. Results indicate that wire tapering reduces delay typically by less than 5% compared to uniform wire sizing, when buffers can be inserted. Consequently, we suggest that it may not be worthwhile to maintain a routing methodology that supports wire tapering.

1. Introduction

Process technology scaling into the deep submicron regime has made interconnect performance more dominant than transistor and logic performance. With the continued scaling of process technology, resistance per unit length of the interconnect continues to increase, capacitance per unit length remains roughly constant, and transistor or logic delay continues to decrease. This trend has led to the increasing dominance of interconnect delay over logic delay, and this trend is expected to continue. The optimization techniques of wire sizing and buffer insertion/sizing have gained widespread acceptance in deep submicron design (see Cong et al. [8] for a survey).

Buffer insertion reduces delay by effectively dividing a wire into smaller pieces, which makes the interconnect delay almost linear instead of quadratic in terms of wire length. Perhaps the best known work in buffer insertion is Van Ginneken's algorithm [20], which finds an optimal solution under the Elmore delay model. Its simplicity and elegance has led to several extensions, e.g., [1][2][13][16].

Widening wire widths can reduce delay by reducing the resistance, but may also increase delay by increasing capacitance. Hence, the optimal wire width depends on the driver strength, interconnect topology, timing constraints, and technology parasitics. Cong et al. [9] first studied wire sizing for signal nets and proposed a dynamic programming approach for a simplistic RC model, but in [10], the authors applied the algorithm to the Elmore delay model [12]. Sapatnekar [19] proposed a sensitivity-based algorithm to handle area and timing constrained formulations. Menezes et al. [14] propose a sequential quadratic programming approach using delays computed by RICE [18]. Kay and Pillegi [12] proposed an efficient, greedy approach.

Chen et al. [3] showed that the optimal wire width under the Elmore delay model is an exponential tapering function $f(x) = ae^{-bx}$ where $a > 0$ and $b > 0$ are constants. The result implies that wire width decreases as the distance x from the driver increases. Chen et al. [4] later considered the same problem while considering fringing capacitance.

Since wire sizing and buffer insertion together affect timing, several works have studied their simultaneous optimization [5][6][7][13][14][16]. Chu and Wong [5] present a closed form solution to the problem for a single line, but do not consider fringing capacitance or buffer sizing. In [6], they propose a quadratic programming approach which handles fringing capacitance, and in [7], they propose an iterative algorithm for a single line. For trees, both Okamoto and Cong [16] and Lillis et al. [13] have proposed wire sizing in conjunction with Van Ginneken's algorithm [20].

The purpose of this work is to analyze the benefits and disadvantages of wire tapering. We use the terms "uniform wire sizing" (UWS) to refer to wire sizing without tapering and "tapered wire sizing" (TWS) to refer to wire sizing with tapering.

Gate sizing is typically performed at both the synthesis and physical design level. All three optimizations (buffers, wires, and gates) are commonly performed after placement but before detailed routing, so Steiner tree estimates are used for the wiring. Tapering could also be performed at this point, as indicated by the works which combine buffer insertion with wire tapering (e.g., [3][4][13][16]). Throughout this work, wire tapering implicitly refers to this post-placement pre-routing stage in the design flow.

Wire tapering can be performed after detailed routing, but routing resources are severely constrained. It is too late to insert buffers and wire widths can only be decreased, not increased. We call this type of tapering "wire shaving".

Currently, tapering is seldom found in typical design flows, and we speculate this is the case for the following reasons:

1. Wire tapering may concentrate routing congestion. For example, consider a 64-bit bus which traverses the entire chip, three bits of which are shown in Figure 1. Wire tapering may widen the wire segments near the drivers which in turn congests this part of the chip. Uniform wire sizing results in narrower widths near the driver and wider widths near the sink. Consequently, congestion will be more evenly distributed.
2. On the other hand, wire tapering may waste area near the sinks (see the dotted lines in Figure 1) unless the router is sophisticated enough to find ways to use the space.
3. Despite advances in gridless routers, most routers still cannot fully exploit tapering. Routing is typically performed (e.g., by the IBM tools Xrouter and LGWire) by

assigning a single *wire code* to each net. The wire code specifies the desired layer and width for each horizontal and vertical segment in the net. In order to support tapering, routers need to handle multiple wire codes.

4. When a wire tapers to a narrower wire, the current density increases. If the current density reaches a certain threshold, electromigration problems can result, and the wire may fuse out. One must be careful to test for these effects or to taper in such a way as to avoid them [11].

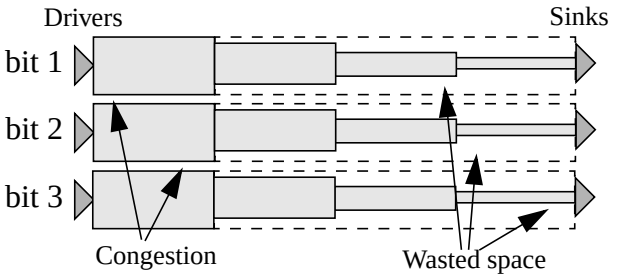

Figure 1 Potential problems that tapering can cause routers.

Although all these problems can be overcome, we hypothesize that it is unnecessary to resort to wire tapering for timing optimization. The combination of buffer insertion, uniform wire sizing, and gate sizing should yield enough delay reduction that any additional reduction from tapering becomes marginal. Our hypothesis[1] is based on the following argument:

- TWS yields significant improvements compared to UWS for very long lines.
- However, the best way to reduce delay for long lines is to simultaneously insert buffers and size wires.
- The insertion of buffers divides the wire from a single long line into several shorter lines.
- For the resulting shorter interconnect segments, TWS will not significantly outperform UWS.

We performed experiments using parasitics for both current and future technologies and discovered that the delay improvement for TWS versus UWS was less than 5% when buffers were inserted. Further, the improvement actually decreases in the more advanced technologies.

2. Problem Formulation

For the purpose of this study, we limit ourselves to single sink as opposed to multi-sink nets for the following reasons:

- For multiple sinks, it is not obvious how to set the required arrival times at each sink.
- The results would be highly dependent on the chosen Steiner topology, and we wish to isolate the effects of wire tapering by reducing the number of variables.
- We are interested in significant delay reduction for long nets. The insertion of buffers on trees will often decouple non-critical sinks thereby reducing the problem to the equivalent of a long line anyway.
- For two pin nets, our heuristic is likely close to optimal which is required to accurately assess the maximum possible benefit of wire tapering. For multi-sink nets, a more sophisticated heuristic would be necessary.

Since UWS is simpler than TWS, we assume that a designer would choose to use TWS only if it were necessary to meet timing constraints that could not otherwise be satisfied. Hence, our objective is to achieve the best possible delay regardless of resource utilization. We focus on discrete wire sizing since the continuous formulation is neither manufacturable nor design rule compatible.

Our formulation assumes a fixed number of buffers. To find the number of buffers which yields the best delay, we iteratively add an additional buffer and re-solve the entire problem until no further delay reduction is possible.

Problem: Given the number of buffers, the number of wire segments between each consecutive pair of buffers, the driver size, the sink capacitance, and the distance from the driver to the sink, optimize delay from the driver to the sink subject to the following conditions:

1. Each wire width can be selected from a fixed set of possible widths $W = \{W_1, \ldots, W_n\}$.
2. Each buffer can be selected from a fixed set of possible buffers $B = \{B_1, \ldots, B_n\}$.
3. Buffers may be placed anywhere on the line between the source and sink.

Figure 2 gives an example. There are 2 buffers and the number of wire segments between consecutive gates is 4, 3, and 4, respectively. If a buffer is shifted left or right, then the lengths of the wire segments connecting the buffer to both the prior and subsequent stage adjust proportionally. Driver sizing is not permitted. Instead, we experiment with a range of driver sizes to examine the effect of this variable.

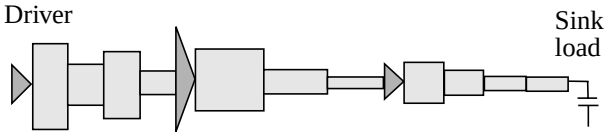

Figure 2 Illustration of the problem formulation.

The following technology parameters are given:

- r_w = sheet resistance.
- c_a = up-down wire capacitance per unit area.
- c_f = fringing wire capacitance per unit length.
- c_{in} = gate input capacitance per unit gate width.
- r_g = gate resistance per unit gate width.
- c_d = gate output capacitance per unit gate width.

We adopt a uniform wire model, i.e., the resistance and capacitance of each wire is uniformly distributed over the area of the wire. The total resistance R_w and capacitance C_w for a wire of width x and length l are given by $R_w = r_w l/x$ and $C_w = l(c_a x + c_f)$, respectively.

[1] Note that our hypothesis does not apply to clocking. In microprocessor design, clock buffers can be much larger than other gates and can drive several millimeters of interconnect without buffering. Unlike signal nets, buffers may not be indiscriminately inserted into a clock tree; hence, clock nets must rely on wire sizing to a much greater degree than signal nets.

Each gate is modeled as a switch-level RC circuit as shown in Figure 3. For a gate of width y, the input capacitance C_{in} and drain capacitance C_d are given by $c_{in}y$ and $c_d y$ respectively. The gate resistance is $R_g = r_g / y$. Instead of Elmore delay, we use AWE [17] using the first four moments of the impulse response to handle the effects of resistive shielding at the driver. Separate experiments verify that our internal delay calculator has accuracy that is consistently within 1% of RICE [18].

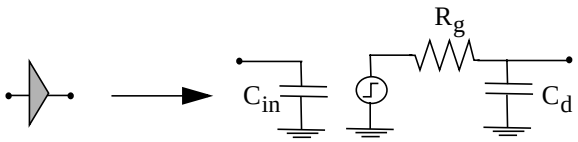

Figure 3 Gate delay model.

3. The Timing Optimization Algorithm

Our objective in developing an algorithm to solve the problem described in Section 2 was not to find an efficient, general or elegant solution, but rather to obtain the best solution possible without regard to CPU time. Since we only wish to assess the maximum possible benefit of wire tapering, the speed of the algorithm is unimportant. Our algorithm (shown in Figure 4) is loosely based on the wire sizing algorithm presented in [12].

Steps 1-3 choose a random solution with randomly sized wires and buffers. From this starting point, the algorithm searches to improve this solution by applying one of four possible optimizations: decreasing a width, increasing a width, shifting a buffer left, or shifting a buffer right. To determine the best move to make, all possibilities are considered and then timed via the internal delay calculator. The move which results in the largest improvement in delay is chosen. If no move improves the delay, then no change is made. Eventually, all four move types will lead to no improvement and the algorithm terminates.

1. Insert buffers so that they are evenly spaced on the line.
2. If performing tapering, segment wires into smaller pieces.
3. Randomly size each buffer and wire.
4. While improving moves remain do
   5. Choose a random number x from $\{1, 2, 3, 4\}$.
   6. If $x = 1$, decrement width of best buffer or wire.
   7. If $x = 2$, increment width of best buffer or wire.
   8. If $x = 3$, shift left the best buffer or wire.
   9. If $x = 4$, shift right the best buffer or wire.

Figure 4 Timing Optimization Algorithm.

Since the algorithm depends on the initial solution, we run it several times and always report the solution with smallest delay. In addition, we implemented an annealing-based optimization algorithm and discovered that the greedy approach generates slightly better solutions; however, the results were different by less than 1% on average. The similarity in results from the annealer versus the greedy approach provide evidence that close to optimal solutions are being generated.

4. Experiments

For our experiments, we used parasitics from four advanced IBM process technologies. We will call these processes A, B, C, and C-SOI. The technologies progress in alphabetic order, so that A is the least advanced, and C is the most advanced technology. With technology C, an SOI or silicon-on-insulator option is available. The interconnect parasitics for C are the same as for C-SOI, but the transistors are faster. For all of the technologies except for A, we also ran experiments for a thick metal layer. These results are denoted by a "T" following the technology, e.g., BT.

The possible wire widths (in microns) were chosen from the set $\{0.9, 1.2, \ldots, 14.1, 14.4\}$. The widths are scaled so that the minimum and maximum widths are the same for each technology and obey the actual design constraints. The possible gate widths (in microns) were chosen from the set $\{10, 20, \ldots, 150\}$, and a 2/1 pfet to nfet ratio was assumed. The possible gate widths and pfet to nfet ratio were chosen to correspond to actual IBM cell libraries [15]. Buffers with width larger than 150 need to be inserted manually.

The following experiments are for an 18 millimeter wire driving a 50 ff load. All delays are reported in picoseconds. The algorithm was run 20 times for each experiment with driver sizes ranging from 10 to 500. Even though drivers of size 250 and 500 are larger than the strongest buffer in our library, we experiment with this size to illustrate the asymptotic effects of increasing driver strength.

4.1 Segmentation Scheme

Our first experiment seeks to find an appropriate number of segments for the remaining experiments. We ran our algorithm with zero buffers for technology B. We repeatedly incremented the number of segments by 10, and optimized the timing until no further delay reduction resulted. As shown in Table 1, timing convergence was always reached by 60 segments, so we use this number for the remaining experiments.

Driver Width	# Segments					
	1	10	20	30	40	50
10	3586.7	3442.6	3441.7	3441.6		
20	2204.8	2074.7	2073.3	2073.0	2072.9	
30	1697.7	1576.7	1575.1	1575.0	1574.8	1574.7
50	1252.7	1143.6	1141.7	1141.2	1141.1	
100	869.2	773.4	770.8	770.5	770.4	770.2
150	720.0	629.8	627.2	626.6	626.4	
250	583.3	497.6	495.3	494.8	494.6	494.5
500	458.8	384.5	382.3	381.9	381.7	381.6

Table 1 Delay after TWS in terms of the number of segments for technology B.

4.2 The Benefit of Tapering without Buffering

By comparing the last value in each column in Table 1 with the first value, one can assess the benefit of TWS versus

UWS, since TWS with 1 segment is equivalent to UWS. For example, for a gate width of 100, UWS yields a delay of 869.2 compared to 770.2 for TWS, which is an 11.4% reduction in delay.

Table 2 shows these delay reduction percentages for TWS versus UWS for each technology. Observe that delay reduction increases significantly with driver strength. For small driver sizes, close to minimum width wires offer the best solution, which means tapering offers little benefit. Stronger drivers cause interconnect delay to dominate, and wire tapering offers significantly more benefit.

Driver Width	Technology						
	A	B	C	C-SOI	BT	CT	C-SOIT
10	3.9	4.0	5.1	5.6	0.3	0.6	0.8
20	5.8	6.0	7.3	7.9	1.9	2.8	3.3
30	7.0	7.2	8.7	9.2	3.2	4.3	4.9
50	8.6	8.9	10.5	11.1	4.8	6.2	6.8
100	11.1	11.4	13.3	14.0	7.1	8.8	9.4
150	12.7	13.0	15.1	15.9	8.5	10.4	10.9
250	14.9	15.2	17.5	18.1	10.4	12.4	13.1
500	16.9	16.8	19.2	19.2	13.1	15.2	15.6

Table 2 Percent improvement of wire tapering over uniform wire sizing for various driver sizes and technologies.

Two other trends can be seen from the table. First, wire tapering appears to help more as technology advances. The same trend can be seen for thick metal as well. This occurs because interconnect resistance increases relative to the driver resistance with advancing technology. Thus, the ratio of interconnect to gate delay increases which makes interconnect tuning, i.e., wire tapering more beneficial. Second, note that for thick metal layers, wire tapering is not as beneficial as for thinner metal. The low resistivity of the thick metal interconnect allows good solutions to have narrower wire widths (often close to minimum width) which minimizes the benefit of wire tapering.

4.3 The Benefit of Tapering with Buffering

The primary conclusion drawn from the previous table is well-known, namely, TWS significantly reduces delay for long lines. We now combine wire sizing with buffer insertion to assess the benefit of TWS in this context.

Table 3 presents the best delays for technology B obtained from 20 runs of our timing optimization algorithm when buffer insertion was combined with wire tapering. We started with 0 buffers and incremented the number of buffers until no further improvement was obtained.

Observe that the first column of the table, which corresponds to full wire tapering with no buffer insertion, is the same as the last entry in each row of Table 1. The best solution for all driver sizes was obtained with 7 buffers or less, so the last entry in each row gives the absolute best delay obtainable by combining both buffering and TWS.

Observe that buffering plus TWS helps significantly. The delay improvement ranges from 8 to 85 percent (as seen by comparing the last column to the first column). Buffering also reduces delay more significantly for weaker drivers since the first few buffers are needed to ramp up the signal strength. Stronger drivers need fewer buffers to achieve the best delay.

Driver Width	# Buffers						
	0	1	2	3	4	5	6 or 7
10	3441.6	791.7	659.4	592.2	551.0	535.2	525.5
20	2072.9	710.3	578.6	536.6	519.6	505.3	499.3
30	1574.7	683.1	551.4	509.4	493.4	488.5	487.3
50	1141.1	641.5	529.4	487.6	471.5	466.7	
100	770.2	551.1	486.6	460.4	450.1	447.7	
150	626.4	494.8	452.2	435.7	430.6		
250	494.5	428.5	406.8	400.3			
500	381.6	355.1	349.8				

Table 3 Best delay obtained by TWS plus full buffering for technology B.

When buffering is combined with TWS, differences in driver strength are not as meaningful. One can see by comparing the last entries in each row, that the range of delays is much smaller than when zero buffers are inserted.

Driver Width	# Buffers						
	0	1	2	3	4	5	6 or 7
10	3586.7	882.9	704.9	632.8	579.9	556.4	539.2
20	2204.8	801.4	623.4	566.0	541.8	527.9	515.9
30	1697.7	770.9	596.2	538.8	515.0	505.3	502.6
50	1252.7	708.1	571.7	517.0	493.1	483.5	481.3
100	869.2	600.9	518.8	484.1	468.5	462.7	462.6
150	720.0	540.0	481.9	457.6	447.7	445.5	
250	583.3	494.8	452.2	435.7	430.6		
500	458.8	397.8	381.4	377.7			

Table 4 Best delay obtained by UWS plus full buffering for technology B.

We repeated the same experiments with uniform wire sizing instead of wire tapering between each buffer stage. The results are shown in Table 4. Observe that the first column of the table is the same as the first column of Table 1.

Observe that the convergence rates in Table 3 and Table 4 are similar. The differences between the tables are much more pronounced for the zero buffer column, than for the maximal buffering. For a driver width of 100, buffering plus TWS results in 447.7 ps delay while buffering plus UWS results in a 462.6 ps delay. Hence, TWS yields a 3.2% reduction in delay over UWS. Compare this to the 11.4% reduction for the same driver size with no buffers inserted.

Buffering plus TWS is not much better than buffering plus UWS for technology B, as improvements range from 2.5 to 3.8 (for drivers of size 150 or less). Table 5 presents the improvements for all of the technologies and driver sizes that we consider. Each entry in the table represents the percent reduction in delay resulting from full buffering and wire tapering versus buffering with uniform wire sizing.

Driver Width	Technology						
	A	B	C	C-SOI	BT	CT	C-SOIT
10	2.6	2.5	3.9	3.1	1.9	2.1	2.5
20	3.5	3.2	3.4	3.6	2.1	2.2	2.6
30	3.6	3.1	3.5	3.0	2.2	2.3	2.6
50	3.8	3.5	3.6	3.0	2.3	2.8	2.7
100	3.5	3.2	3.3	3.3	2.9	2.7	2.6
150	4.3	3.8	3.3	3.3	2.6	3.1	2.9
250	6.1	5.2	4.1	4.0	5.9	4.5	4.1
500	10.0	8.3	7.0	6.5	12.9	10.1	8.4

Table 5 Percent improvement of TWS over UWS with maximal buffering for an 18 mm wire.

- For driver sizes with width of 150 or less, the improvement of TWS over UWS is 4.3% or less. Only once the driver exceeds 150 does the improvement exceed 4.3%.
- The improvement generally decreases as technology advances (from A to B to C to C-SOI). More buffers are required which reduces the wire length between subsequent buffers, which in turn, reduces the impact of tapering. For a driver size of 100, the optimal buffering and wire tapering solution had 4, 5, 7 and 8 buffers for technologies A, B, C, and C-SOI, respectively.
- The delay reduction from wire tapering and full buffering for thick metal is actually less (under 3.1%) than for regular metal until driver widths greater than 150 are used. For thick metal and smaller drivers, the optimal wire widths are typically close to minimum, which gives TWS little opportunity for improvement over UWS

4.4 Other Wire Lengths

To see if tapering had a potentially different effect on significantly shorter or longer wires, we performed the same experiments for both 6 and 30 millimeter wires. The percent improvements of TWS over UWS with maximal buffering are shown in Table 6 and Table 7, respectively.

For the 6 (30) mm wire, TWS always yields less than 7.7 (3.8)% improvement versus UWS, when the driver width is 150 or less. The largest improvements are actually for the shortest wire. Improvements are as high as 15.0% for the 6 mm wire compared to 12.9% and 7.9% for the 18 and 30 mm wires. This phenomenon can be explained by the fact that the driver size is more significant in terms of the number of buffers for shorter wires.

From the tables, one can extrapolate that as the driver size increases beyond 500, the improvement of TWS over UWS will increase as well. However, a width of 150 already pushes reasonable design limits; in practice, drivers with width less than 150 are typical.

Driver Width	Technology						
	A	B	C	C-SOI	BT	CT	C-SOIT
10	3.7	2.0	3.2	2.3	2.0	1.5	1.8
20	4.2	2.3	3.9	2.6	2.4	1.7	2.1
30	4.9	2.8	3.9	2.8	2.8	2.0	2.3
50	5.6	2.9	4.2	3.1	3.2	2.2	2.6
100	4.2	4.5	3.5	3.9	2.7	3.3	2.3
150	7.7	4.0	5.3	3.7	4.5	2.8	3.2
250	9.3	9.7	5.8	6.4	5.9	7.5	8.1
500	11.7	12.2	14.2	15.0	8.0	9.9	10.5

Table 6 Percent improvement of TWS over UWS with maximal buffering for a 6 mm wire.

Driver Width	Technology						
	A	B	C	C-SOI	BT	CT	C-SOIT
10	2.8	3.3	3.5	2.9	2.8	2.8	3.3
20	3.3	2.7	3.2	3.0	4.1	3.6	3.0
30	3.1	3.1	3.3	3.0	2.5	2.6	3.1
50	3.4	3.2	3.4	3.0	2.6	2.6	2.7
100	3.3	3.0	3.0	3.2	2.5	2.5	2.6
150	3.8	3.3	3.3	3.5	2.8	2.8	2.9
250	4.7	4.0	3.8	3.7	3.9	3.5	3.5
500	6.7	5.6	4.8	5.0	7.9	6.1	5.8

Table 7 Percent improvement of TWS over UWS with maximal buffering for a 30 mm wire.

4.5 Resource Comparisons

We speculated in Section 1 that a possible disadvantage of tapering was increased congestion near the driver. To see if this occurs, we compare the maximum wire widths and areas generated by our algorithm for TWS and UWS. Table 8 shows the results for Technology B for both full buffering and for no buffering. Area is reported in square microns (x 10^3) and the maximum width is reported in microns.

From the table, one can see that with no buffering, UWS uses slightly more area than TWS, but has significantly smaller maximum width. Thus, more congestion may result near the drivers when TWS is employed.

With full buffering, the results are similar. The areas are about the same for TWS and UWS while the maximum wire width is again higher for TWS. Overall, the maximum wire widths are smaller as compared to no buffering, which is due to the shorter wire lengths offering less opportunity for sizing. Neither approach has a definitive area advantage.

4.6 Optimization Trade-offs

Table 9 shows the contribution towards delay reduction of

buffer insertion and wire sizing for technology B. The first three data columns are for no buffering and the last three are for full buffering.

Driver Width	No Buffering				Full Buffering			
	Area		Max Width		Area		Max Width	
	UWS	TWS	UWS	TWS	UWS	TWS	UWS	TWS
10	32.4	30.2	1.8	2.4	61.9	61.4	3.6	4.5
20	43.2	42.5	2.4	3.6	64.4	64.2	3.9	4.8
30	54.0	51.5	3.0	4.5	61.0	61.5	3.6	4.5
50	64.8	66.6	3.6	6.0	65.4	63.9	3.9	4.8
100	91.8	95.0	5.1	9.3	61.6	62.9	3.6	4.8
150	108.0	116.6	6.0	11.7	63.6	64.5	3.9	4.8
250	140.4	148.9	7.8	14.4	77.0	76.6	5.7	9.0
500	189.0	171.4	10.5	14.4	108.0	117.2	8.7	14.4

Table 8 Area and max width comparisons of UWS versus TWS for technology B.

Driver Width	No Buffering			Full Buffering		
	None	UWS	TWS	None	UWS	TWS
10	3880.4	3586.7	3441.6	719.4	539.2	525.5
20	2673.6	2204.8	2072.9	689.0	515.9	499.3
30	2251.3	1697.7	1574.7	674.0	502.6	487.3
50	1899.8	1252.7	1141.1	656.7	481.3	466.7
100	1624.3	869.2	770.2	632.8	462.6	447.7
150	1529.3	720.0	626.4	620.5	445.5	430.6
250	1452.0	583.3	494.5	608.5	430.6	400.3
500	1393.0	458.8	381.6	597.7	377.7	349.8

Table 9 Delay comparisons for technology B.

Comparing the UWS and TWS columns with no buffering with the None column for full buffering, observe that buffering without wire sizing is significantly better than any wire sizing when the driver size is 100 or less. For example, for a gate width of 50, buffering without wire sizing yields 656.7 ps of delay, as compared to 1252.7 and 1141.1 for UWS and TWS without buffering. Comparing the None and UWS columns for full buffering, we see that UWS consistently shaves 170-200 ps off the buffered solution without wire sizing. Additional wire tapering shaves off another 14-30 ps. Compared to the large reductions achieved by only buffering, and the subsequent medium sized reductions achieved by UWS, the additional delay reductions achieved by TWS is only a third order effect.

5. Conclusion

Timing optimization is a critical step in modern-day design, and buffer insertion and wire sizing are key steps in this process. Wire tapering for signal nets can cause headaches, such as routing congestion, poor track utilization, and routing time complexity for current design flows. We have tried to address the question, “Is it worthwhile to perform wire tapering?”

Our studies suggest that it is not worthwhile to perform wire tapering on a large number of global signal nets. This conclusion is based on experiments which show that TWS typically leads to less than 5% reduction in delay versus UWS when nets are maximally buffered.

References

[1] C. J. Alpert, A. Devgan, and S. T. Quay, “Buffer Insertion for Noise and Delay Optimization”, *35th IEEE/ACM Design Automation Conference*, 1998, pp. 362-367.

[2] C. J. Alpert, A. Devgan, and S. T. Quay, “Buffer Insertion with Accurate Gate and Interconnect Delay Computation”, *36th IEEE/ACM Design Automation Conference*, 1999, pp. 479-484.

[3] C.-P. Chen, Y.-P. Chen, D. F. Wong, “Optimal Wire-sizing Formula under the Elmore Delay Model”, *33rd IEEE/ACM Design Automation Conference.*, 1996, pp. 487-490.

[4] C.-P. Chen, Y.-P. Chen, D. F. Wong, “Optimal Wire-sizing Function with Fringing Capacitance Consideration”, *34th IEEE/ACM Design Automation Conf.*, 1997, pp. 604-607.

[5] C. C. N. Chu and D. F. Wong, “Closed Form Solution to Simultaneous Buffer Insertion/Sizing and Wire Sizing”, *Intl. Symposium on Physical Design*, 1997, pp. 192-197.

[6] C. C. N. Chu and D. F. Wong, “A New Approach to Simultaneous Buffer Insertion and Wire Sizing”, *IEEE/ACM Intl. Conference on Computer-Aided Design*, 1997, pp. 614-621.

[7] C. C. N. Chu and D. F. Wong, “A Polynomial Time Optimal Algorithm for Simultaneous Buffer and Wire Sizing”, *Conf. Design Automation and Test in Europe.*, 1998, pp. 479-485.

[8] 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.

[9] J. Cong, K. S. Leung, and D. Zhou, “Performance-driven Interconnect Design Based on Distributed RC Delay Model”, *IEEE/ACM Design Automation Conf.*, 1993, pp. 606-611.

[10] J. Cong and K. S. Leung, “Optimal Wiresizing under the Distributed Elmore Delay Model”, *IEEE/ACM International Conference on Computer Aided Design*, 1993, pp. 634-639.

[11] G. Ellis, *personal communication*, March 1999.

[12] R. Kay and L. T. Pileggi, “EWA: Efficient Wiring-Sizing Algorithm for Signal Nets and Clock Nets”, *IEEE Transactions on Computer-Aided Design*, 17(1), 1998, pp. 40-49.

[13] J. Lillis, C.-K. Cheng and T.-T. Y. Lin, “Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model”, *IEEE J. Solid-State Circuits*, 31(3), 1996, 437-447.

[14] N. Menezes, R. Baldick, and L. T. Pileggi, “A Sequential Quadratic Programming Approach to Concurrent Gate and Wire Sizing”, *IEEE/ACM Intl. Conf. on CAD*, 1995, pp. 144-151.

[15] G. Northrop, *personal communication*, Feb. 1999.

[16] T. Okamoto and J. Cong, “Buffered Steiner Tree Construction with Wire Sizing for Interconnect Layout Optimization”, *IEEE/ACM Intl. Conf. Comp. Aided Design*, 1996, pp. 44-49.

[17] L. T. Pillage and R. A. Rohrer. Asymptotic Waveform Evaluation for Timing Analysis. *IEEE Transactions on Computer-Aided Design.* 9(4), 1990, pp. 352-366.

[18] C. Ratzlaff and L. T. Pillage, “RICE: Rapid Interconnect circuit Circuit Evaluator using Asymptotic Waveform Evaluation”, *IEEE Trans. Comp.-Aided Design*, 763-776, June 1994.

[19] S. S. Sapatnekar, “RC Interconnect Optimization under the Elmore Delay Model”, *31st IEEE/ACM Design Automation Conference*, 1994, pp. 387-391.

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