Spec-based Repeater Insertion and Wire Sizing for On-chip Interconnect

Noel Menezes, Chung-Ping Chen
Strategic CAD Labs, Design Technology
Intel Corporation
Hillsboro, OR 97124

Abstract

Recently Lillis, et al. *presented an elegant dynamic programming approach to RC interconnect delay optimization through driver sizing, repeater insertion, and, wire sizing which employs the Elmore delay model for RC delay estimation and a crude repeater delay model. This approach, however, ignores an equally important aspect of interconnect optimization: transition time constraints at the sinks. More importantly, Elmore delay techniques because of their inherent inaccuracy are not suited to spec-based design which is directed towards synthesizing nets with user-specified delay/transition time requirements at the sinks. In this paper we present techniques for delay and transition time optimization for RC nets in the context of accurate moment-matching techniques for computing the RC delays and transition times, and an accurate driver/repeater delay model. The asymptotic increase in runtime over the Elmore delay model is $O(q^2)$ where q is the order of the moment-matching approximation. Experiments on industrial nets indicate that this increase in runtime is acceptable. Our algorithm yields delay and transition time estimates within 5% of circuit simulation results.*

1 Introduction

Interconnect is fast becoming a major limiter to CMOS IC performance due to aggressive scaling of processes. Consequently, physical design in addition to its traditional density-oriented goal of area minimization needs to be directed towards eletrical objectives like power and delay reduction which have been usually associated with the circuits domain. Concurrent driver and wire-sizing [4, 5, 8, 16] has been shown to be a powerful technique for interconnect delay reduction. However, repeater insertion is widely recognized as the most effective technique for delay and transition time improvement in RC nets.

The effect of repeater insetion on the delay of a net is well documented. Essentially, the delay of an RC line scales quadratically with its length. With repeater insertion this relationship becomes a linear one which greatly alleviates the interconnect problem with current technologies. Another equally advantageous but frequently ignored effect of repeater insertion in large RC nets is the signal restoration that CMOS repeaters provide due to their high-gain amplifier nature. Since the lowpass nature of RC nets severely degrades the transition times of the signals at the sinks, inserting a repeater in such nets has the effect of improving the transition times. Nets with large transition times are not desirable for several reasons: primarily for reliability reasons as well as as their poor noise immunity. In several cases during design even when the delay of a net is acceptable the transition time of the degraded signals at the sinks may necessitate repeater insertion. *Consequently, it is imperative that any interconnect optimization solution must take transition times into account.*

Recently, in [6] Lillis, *et al*. presented an elegant dynamic-programming approach to concurrent repeater insertion, driver, and wire sizing for RC nets which is based on an earlier algorithm for repeater insertion proposed by Ginneken [7]. Essentially, Lillis exploited the hierarchical nature of the Elmore delay model to construct a set of feasible solutions in a bottom-up manner. Effective pruning techinques were used to poynomially-bound the size of the solution set while traversing the tree from the sinks to the root. A major drawback of Lillis' technique is the inability to account for the resistive shielding of the RC net [13]: the driver delay for an RC net is significantly smaller than that predicted using the total capacitance of the net due to the resistance in the load.

While the Elmore delay is a reasonable estimator of the RC delay its accuracy is limited. Another major drawback of the Elmore delay model and, therefore, Lillis' technique, is its inability to accurately estimate transition times. Hence, while the Elmore delay model is applicable in certain scenarios (a generic minize-maximum-delay formulation) its usage in high-frequency designs leads to sub-optimal solutions. For example, during spec-based design when the requirements are usually specified in terms of a delay and transition-time constraint a net optimized using the Elmore delay (which is known to be conservative) may be "overdesigned" -- that is, the post-optimization net could exceed its design goals when simulated -- which implies a needless power and/or area cost. In this paper we present efficient techniques for removing this conserva-

tism both in delay and transition-time estimation during interconnect optimization through the usage of more accurate moment-matching techniques like Asymptotic Waveform Evaluation (AWE) [17] and Pade-via-Lanczos (PVL) [10] for a reasonable increase in computation time. Our basic algorithm follows the Ginneken/Lillis dynamic programming framework. The main contributions of this paper are:

- In addition to delay, transition time constraints which are imperative for interconnect optimization are handled.
- The effect of the input transition time on the repeater delay and transition time is taken into account
- Moment-matching techniques are used for accurate RC delay estimation: Efficient bottom-up computation of the RC delay is based on a novel application of the REX hierarchical moment computation technique proposed in [11].
- An accurate repeater/driver model [12] that takes resistance-shielding of the RC net is into account in addition to providing accurate waveform estimates at the sinks.

To the best of our knowledge this is the first work in interconnect optimization that handles these real-world constraints through the use of advanced delay models for both the RC as well as the driver/repeater components.

2 Algorithmic framework

We use the following notation in this paper:

- T: A routing tree with n branches and a set of s sinks $\{R_1, R_2, ..., R_s\}$ (receivers).
- e_i: i-th branch of the tree, $1 \le i \le n$.
- l_i: Length of e_i.
- $Children(e_i)$: Set of the immediate children of e_i.
- W_i: The set of feasible wire sizes for e_i.
- $rcdelay(e_i, w_i, c)$: The RC delay of branch e_i sized to width $w_i \in W_i$ when loaded by a capacitive load c.
- $cap(e_i, w_i)$, $res(e_i, w_i)$: Capacitance and resistance of branch e_i when sized to width $w_i \in W_i$.
- B: Library of repeaters.
- c_b: Input capacitance of a repeater $b \in B$.
- $delay(b, c)$: Delay of repeater b when loaded by a capacitive load c.
- T_i: User-specified maximum delay constraint at sink R_i.
- C_i: Capacitance load at sink R_i.

An example routing tree is given in Figure 1.

The framework of the Lillis/Ginneken algorithm is outlined in Figure 2. As illustrated in Figure 1b we insert repeaters only at the end of the branch away from the root of the tree. Starting at the receivers and moving towards the root, at the end point of each routing segment e_i a repeater is inserted in turn from the set of allowable repeaters B_i for segment e_i, and a set of solutions constructed in terms of the set of solutions of the branches

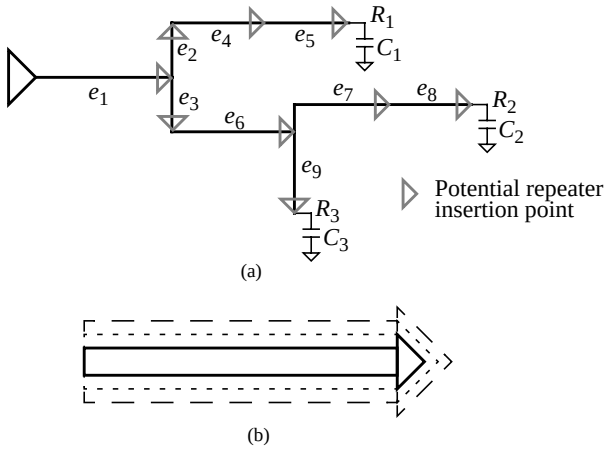


Figure 1: (a) A routing tree with potential repeater location at the non-root-oriented end of each branch (b) A branch of the routing tree with different possible width configurations and a set of repeater choices.

immediately downstream of this point. The solution set at each insertion point consists of a set of (load, required time) — (c, t) — pairs each of which corresponds to a partial solution for the repeater insertion problem up to this point. The quantity c corresponds to the input capacitance for this solution seen from the insertion point. The quantity t represents the required time at this insertion point. The information encapsulated in each pair is used for recursively constructing the set of solutions at the insertion point immediately upstream of this insertion point. A very effective pruning technique is then used to remove obviously non-optimal pairs from this set. At the driver the final set of solutions is constructed by backtracking from the set of pairs generated at the driver

2.1 Algorithm description

This technique is more formally described by the following recursion:

Base case: At each sink R_i with load capacitance, C_i, and required time, T_i, the solution set, S_i, consists of a single pair: (C_i, T_i) (lines A1 and A2 in Figure 2). The load for this pair is the input capacitance of the sink while the required time of this pair is the user-specified required time for sink R_i.

Induction case: Assume that we have the solution set, S_j, of pairs at a particular insertion point j (Figure 3a). We need to construct the solution set for the edge e_i immediately upstream of j.

Repeater insertion**:** (Lines A7 through A9)The first step is to incorporate the effects of repeater insertion at j into the set of solutions, S_j. For every repeater $b \in B$ with input capacitance, c_b, we add a new pair P to S_j. This pair is constructed by searching all pairs of S_j in order to find the pair which will yield the latest required time when repeater b is inserted at point j. The load for this new pair P will be the

InsertRepeaters(e_i)
A1 **if** e_i is a sink element
A2 $S_i = \{ (C_i, T_i) \}$
A3 **return**
/* merge children */
A4 $S_i = \{ \}$
A5 **foreach** $e_j \in Children(e_i)$
A6 S_i = Merge (S_i, S_j)
/* insert repeaters */
A7 **foreach** $b \in B_i$
A8 find $(c, t) \in S_i$ such that
$$t - delay(b, c) > t' - delay(b, c') \quad \forall (c', t') \in S_i$$
A9 $S_i = S_I \cup \{(c_b, t - delay(b, c))\}$
A10 Prune(S_i)
/* wire sizing on e_i */
A11 **foreach** $w \in W_i$
A12 **foreach** $(c, t) \in T_i$
A13 $(c, t) \leftarrow (c + cap(e_i, w_i), t - rcdelay(l_i, w_I, c))$
A14 Prune(S_i)

Merge(S_i, S_j)
/* S_i, S_j are ordered in terms of c */
A15 $S = \{\}$
A16 $(c_i, t_i) = pop(S_i)$, $(c_i, t_i) = pop(S_2)$
A17 **while** $(c_i, t_i) \neq NULL$ **and** $(c_j, t_j) \neq NULL$
A18 $S = S \cup \{(c_i + c_j, min(t_i, t_j))\}$
A19 $t_i' = t_i$, $t_j' = t_j$
A20 **if** $t_i' \leq t_j'$ **then** $(c_i, t_i) = pop(S_i)$
A21 **if** $t_j' \leq t_i'$ **then** $(c_j, t_j) = pop(S_j)$
A22 **return** S

Figure 2: Lillis/Ginneken algorithmic framework for repeater insertion and wire sizing.

input capacitance of the repeater. The required time for the new pair will be the required time for the latest required time pair decreased by the delay, *delay*(*b*, *c*) of the repeater for the load *c* (Figure 3a).

Wire sizing: (Lines A11 through A13) Finally the solution set S_i is constructed by sizing branch e_i and incorporating the RC delay of the sized segment e_i for each pair of S_j. For each possible wire width w_i of e_i, each pair (*c*, *t*) is transformed into a new pair (*c* +*cap*(e_i, w_i), *t* - *rcdelay*(e_i, w_i, *c*)) which accounts for the additional load *cap*(e_i, w_i) added to the load of the pair and the decrease in the required time due to the RC delay, *rcdelay*(e_i, w_i, *c*), of segment e_i (Figure 3b).

Merging of branches: (Line A5 and A6) When a merging point is reached during the backward traversal along the forking branches of the net, the solution sets for each

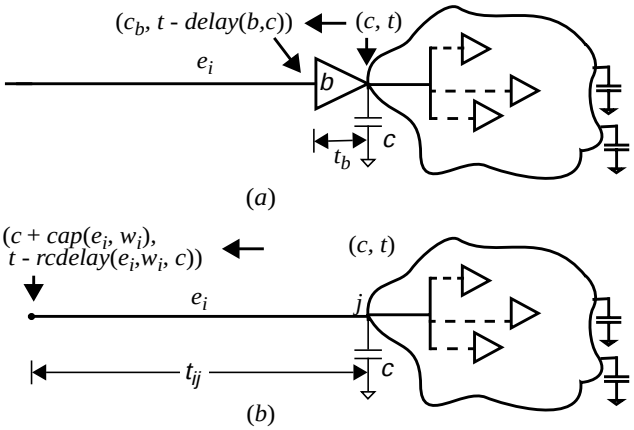


Figure 3: Hierarchical pair generation: a) repeater *b* inserted at *j*, b) no repeater inserted at *j*.

branch are merged at the forking node using a simple set of rules described in the Merge() routine of Figure 2.

When the backward traversal reaches the driver, each pair belonging to the solution set at the driver node with a positive required time describes a feasible solution to the repeater insertion and wire sizing problem.

2.2 Ginneken's pruning lemma

It is easy to see that this is a potentially exhaustive technique since the size of the solution set geometrically increases by the number of repeaters in the library for every insertion point from the receiver to the driver. At each insertion point after a solution set has been created we prune inferior pairs using the following rule suggested by Ginneken [7] which reduces the size of the solution set drastically:

> For any two pairs, (c, t) and (c', t'), in a solution set *S*, if $c < c'$, and $t > t'$, then the pair (c', t') can be removed from set *S*.

As for the runtime, Ginneken points out that the solution set size is bounded by the number of possible distinct load values. Thus, while there is an exponential number of buffer and wire size assignments, there is a polynomial number of distinct resulting loads and relevant solutions. The upper bound on the total runtime is $O(n^3 c_{max}^2 \log(nc_{max})(|B| + |W|))$ where c_{max} is the largest discretized capacitance possible for the tree [6].

3 Delay modeling

3.1 Accurate driver/repeater modeling

With scaling of processes to the deep submicron domain the transistor characteristics have been improving at a much better rate than the interconnect resulting in a reduced on-resistance of the transitor but an increase in the relative value of the interconnect resistance. As a result the resistance of the interconnect has become significant com-

pared to the driver output resistance. For average nets it has been empirically determined that the load presented by an RC net to its driver is much better represented by a π-model synthesis of its admittance than the total capacitance load approximation used hitherto. This is because the resistance of the net shields a part of the load capacitance from the driver which results in a smaller driver delay than that predicted by lumping the total capacitance of the net into a single capacitive load. Hence, conventional delay models which do not account for resistance shielding significantly overestimate the driver delay.

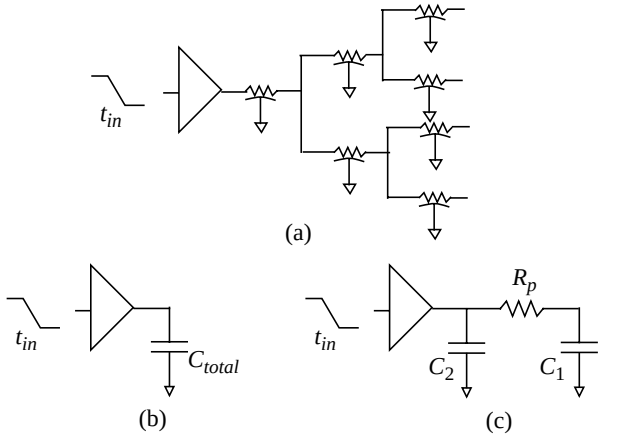


Figure 4: a) A driver loaded by an RC net, b) first-order total capacitance representation c) second order π-model representation.

This phenomenon is illustrated in Figure 5a where the waveforms obtained for the two different load approximations for a particular driver-net combination are shown. The π-load waveform overlaps the actual waveform (not shown) almost exactly. The parameters of the π–load which is a second-order approximation of the driving point admittance $Y(s)$ (a single capacitor is a first-order approximation) can be trivially computed in linear time for an RC tree by the algorithm described in [15]

We use the delay model of [12] which can provide accurate driver delays in the presence of π-loads. The inputs to this driver model are a driver precharacterized in terms of the input transition time, t_{in}, and output load, C_L and, the π-load of the RC net. The delay model of [12] replaces the driver by a resistor and a voltage source. The parameters of the voltage source are determined by iterative procedure which equalizes the average current drawn by the driver for a π-load and a purely capacitive load which mimics the "effective capicitance" of the π-load.

In our algorithm, the load of the RC net which is expressed by the π-model is hierarchically computed during the bottom-up phase and stored with the pair data structure. That is, each pair (c, t) is tagged with an admittance field $Y(s)$. The computation of the admittance and the transfer function (which is required for accurate RC delay computation) for each pair is described in the next section.

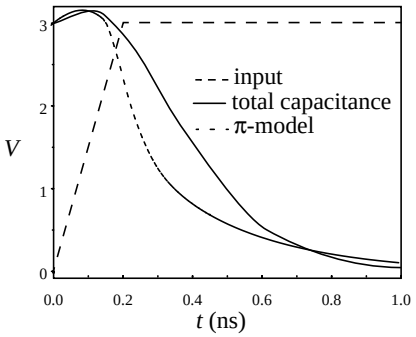


Figure 5: Driver output waveforms for a π-load and a total capacitance load,

3.2 Hierarchical moment computation for accurate RC delay computation

An assumption made during the explanation of the required time computation for pair P in Section 2.1 is that the delay from i to a receiver (see Figure 6) can be split into three additive components: t_{ij}, the RC delay, t_{jk}, from j to some repeater k downstream of j, and the delay from this repeater to a downstream receiver. This hierarchical delay computation is possible under the Elmore delay model which does not take signal waveshapes into account. However for a more accurate model the RC delays are not additive, that is, $t_{ij} + t_{jk} \neq t_{ik}$ since these quantities depend largely on the waveshapes of the signals at nodes i and j. So instead of hiearchically computing the delay a more accurate way is to compute the transfer function hierarchically and then compute the delay by convolution of the input signal waveshape with the composite transfer function.

A significant innovation introduced in this paper is the use of hierarchical moment computation based on Asymptotic Waveform Evaluation (AWE) [17] for accurate RC delay computation. Moment-matching techniques work on the principle of matching the first $2q$ moments of the transfer function from the root to any node in an RC tree:

$$H(s) = m_0 + m_1 s + m_2 s^2 + \ldots + m_{2q-1} s^{2q-1} \qquad (1)$$

to that of a reduced-order pole-and-residue representation:

$$\tilde{H}(s) = \sum_{i=1}^{q} \frac{\tilde{k}_i}{s - \tilde{p}_i}. \qquad (2)$$

A 3rd-order ($q = 3$) approximation is sufficient to capture a reasonably accurate response for RC nets with thousands of capacitors! Furthermore, the first $2q$ moments for all n nodes of an RC tree can be computed in $O(2qn)$ time. (The Elmore delay is the first moment of the transfer function.)

For the purposes of repeater insertion, moment matching can be applied to the problem of calculating the RC delay of the partial subtrees during the bottom-up phase. Under the Lillis/Ginneken framework, this would require

that for each pair we trace back to the first repeaters and construct the RC subtree, and then analyze this subtree using moment matching techniques. A back-of-the-envelope calculation for this shows that in spite of the speedup afforded due to AWE-like techniques, the runtime cost would be prohibitive because of the large number of partial subtrees that need to be analyzed during the bottom-up dynamic programming algorithm! Ideally, we would like to use a technique that preserves the hierarchical nature of the Elmore delay. That is, in order to compute t_{ik} we would like to be able to use the information generated during the computation of t_{jk}.

In our approach, in order to calculate t_{ik}, we compute the moments of the transfer function, $H_{ik}(s)$, from the electrical parameters of the branch e_i and from the moments of $H_{jk}(s)$ and admittance $Y_j(s)$ which are part of the data structure of the pair under consideration at node *j*. The electrical model for the computation of $H_{jk}(s)$ and $Y_i(s)$ are shown in Figure 6b for which the formulae are:

$$\begin{aligned} H_{ij}(s) &= \frac{1}{R(Cs + Y_j(s)) + 1} \\ Y_i(s) &= Cs + \frac{Cs + Y_j(s)}{R(Cs + Y_j(s)) + 1} \\ H_{ik}(s) &= H_{ij}(s)H_{jk}(s) \end{aligned} \tag{3}$$

where R = $res(e_i, w_i)$ and C = $cap(e_i, w_i)/2$. The RC delay t_{ik} is then computed by convolution of the waveform at *i* and $H_{ik}(s)$ (Figure 6). The moments of $H_{ik}(s)$ and $Y_i(s)$ are

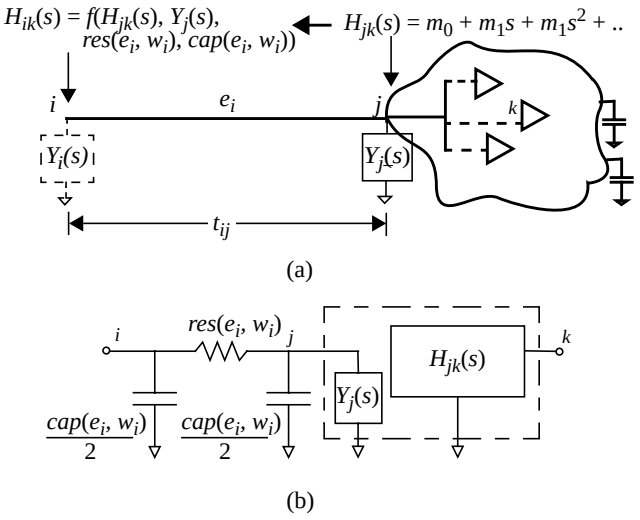


Figure 6: Hierarchical moment computation a) transfer function and admittance propagation b) equivalent electrical model

then added to the data structure of the pair at node *i*. The π-model parameters required in order to determine the repeater delays using the model of Section 3.1 at node *i* can be computed from the first three moments of the admittance $Y_i(s)$ using the formulae of [15].

The hierarchical moment generation for the transfer function and input admittanec always starts from either a receiver or a repeater. For this base case the moment representation of the transfer function is given by:

$$\begin{aligned} H(s) &= 1 \\ Y(s) &= cs \end{aligned}. \tag{4}$$

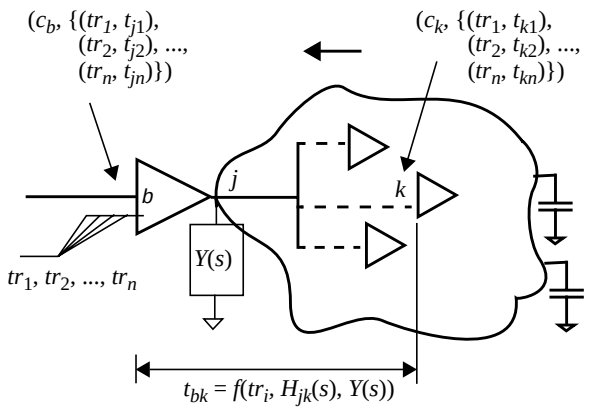
The induction case is illustrated in Figure 6 where a certain transfer function and admittance representation are "stitched" together with a segment e_i. The formulas for this stitching are given in (3). The transfer functions and admittance are always represented in their moment form of (1) in the pair data structure. When the delay and transition time computation is required the transfer function is converted into the poles-and-residues representation of (2) using Asymptotic Waveform Evaluation [17].

4 The effect of input transition times

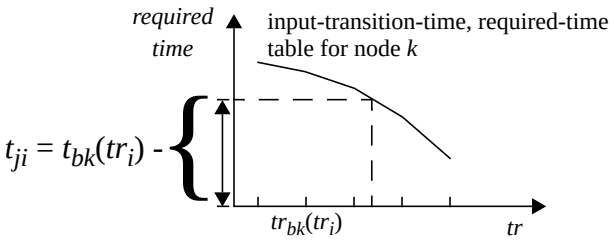
The delay and output transition time of a repeater, and consequently the delay of the RC net that it drives, is a strong function of the transition time of the input signal. We describe how we deal with this important effects:

In a previous section we described the repeater delay model that takes resistance shielding effects into account. One of the inputs to this model is the input transition time of the repeater. Hence, in Figure 3a, while inserting a repeater at point *j*, we need to take the transition time of the input signal into account. This transition time, however, is unknown because of the bottom up nature of this algorithm. Hence, in practice for each repeater we do not generate a pair (c_b, $t - t_b$) with a single required time but instead we generate a pair that includes a table of (input-transition-time, required time) pairs (c_b, {(tr_1, t_1), (tr_2, t_2), ..., (tr_n, t_n)}) for *n* possible input transition times — tr_1, tr_2, ..., tr_n. When this pair is propagated backwards through a repeater upstream of the current repeater we index into this table to determine the required time for the repeater upstream based upon the transition time of the previous stage. This process is illustrated in Figure 7.

For example, while inserting a repeater at node *j*, we find the best pair for this repeater as described on lines A7-A8 of the algorithm in Figure 2. As described in the previous section this pair carries with it the moments of the input admittance of the partial subtree, $Y(s)$, as well as the moments of the transfer function, $H_{jk}(s)$. Given this information, for each value of the input transition time tr_i we determine the delay from the input of the repeater *b* to the node *k*, t_{bk}, using the repeater delay model of Section 3.1 and moment matching techniques [12]. During this process we also compute the transition time at node *k*, tr_{bk}. The required time at node *j* for the input transition time tr_i is then determined from t_{bk} and linear interpolation into the (input-transition-time, required-time) table of node *k* (Figure 7b).



(a)



(b)

Figure 7: Handling of input transition time effects: (a) delay model for insertion at node *j*, (b) computation of required time.

4.1 Handling of transition time constraints

Given the background that we have developed in the previous section transition time handling is trivially explained. Each entry of the (input-transition-time, required-time) table is also tagged with the output transition time. When creating this table for the first repeater before a sink if it turns out that the output transition time is less than the user-specified constraint then this entry is eliminated from the table. Thereafter during bottom-up propagation if it turns out that the output transition time exceeds that of any entry in the table then that input transition time is not valid. For example, in Figure 7 if for a certain input transition time tr_i to the repeater at node *j* the output transition time at node *k* is greater than the transition time associated with any of the entries in the table for node *k* then that entry for tr_i (and those greater than tr_i) is removed from the table for node *j*.

The possibility that the critical sink (or repeater) from a delay point-of-view (that is the sink with the latest required time) is not the one where the worst transition time is observed is handled in a straightforward manner. Furthermore, in order to avoid pruning potential optimal solution which has a worse delay but a better transition time, we also add a transition time factor into the dynamic programming table. Hence, during pruning (Step A8), we add an extra condition to ensure that we only eliminate solutions that are inferior from a transition time point of view as well:

> For any two pairs, (c, t) and (c', t'), in a solution set *S*, if $c < c'$, and $t > t'$, and *transition_time*$(b, c) <$ *transition_time*(b, c') then the pair (c', t') can be removed from set *S*.

where *transition_time*(b, c) is the transition time at the output of repeater *b* for a capacitive load *c*.

We assert here that a transition-time-driven repeater insertion algorithm will suffer from significant accuracy problems if these effects are not taken into account.

5 Experimental results

We have run our implementation of the algorithm described above on thousands of nets taken from a previous-generation microprocessor. These nets had a demanding transition time requirement in addition to the user-specified delay requirement. We use a library of inverters of different sizes as repeaters. A comparison of our algorithm for a set of nets that were manually designed by circuit designers for the same delay and transition time constraints is shown in Figure 8. We can see that our algorithm was able to optimize the delay of these nets to a value very close to those achieved by hand design. More importantly, in all these case the transition time requirements was met with a <5% error. Also shown are the delays predicted by our algorithm compared to the delay predicted by circuit simulation (SPICE) for the optimized nets.

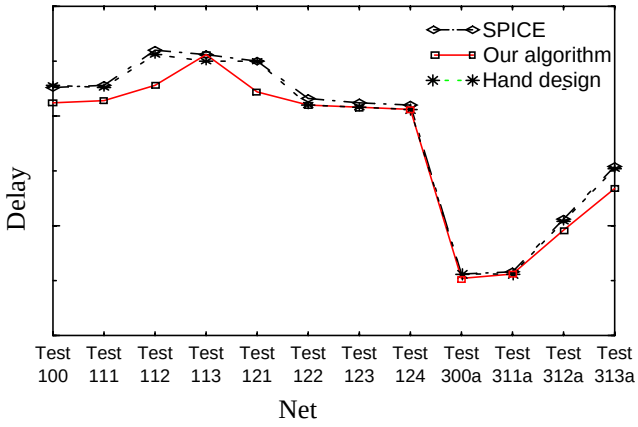


Figure 8: Comparison of nets optimized using our algorithm against the manually designed nets. (The delay axis units have been intentionally omitted.)

In Figure 9 the effect of tightening the transition time requirements for repeater insertion along a line is shown. We insert repeaters into a 10500 µm net by breaking it into 500 µm segments for both a 1 time unit and a 0.4 time unit transition-time requirement respectively. Not surprisingly, it can be seen that a more aggressive transition time requirement results in a larger number of repeaters. The runtime required for optimizing this net was on the order

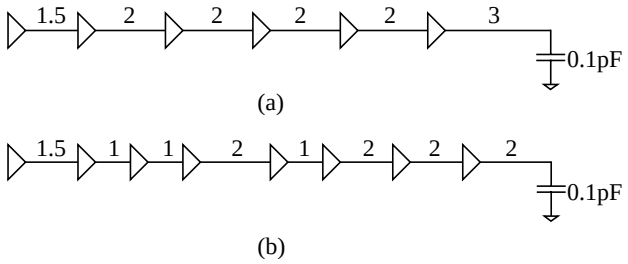


Figure 9: Repeater insertion for a 10500 μm line for a (a) 1 time unit, (b) 0.4 time unit transition-time constraint. (All lengths are in mm.)

of 2 seconds on an IBM RS6000 workstation.

Finally, we present a particularly challenging net with 394 segments. The figure shows a screen capture from an interface that displays the results of repeater insertion. The large diamond represents the driver while the small diamonds show the repeater insertion location. The circles represent the sinks. Repeater insertion for this net took 67 cpu-seconds and yielded a 20-repeater solution.

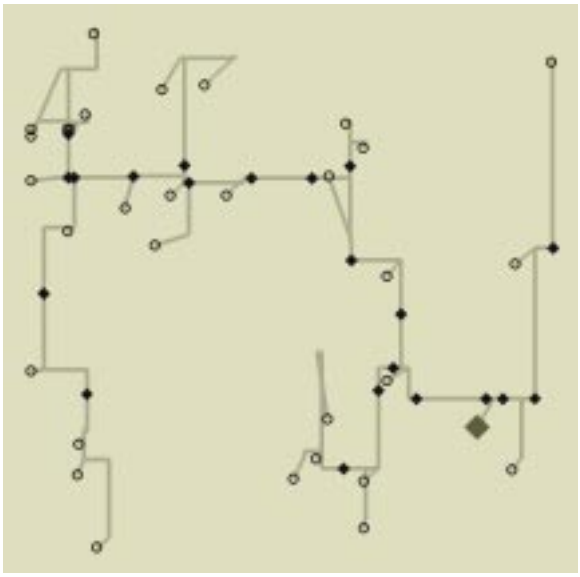


Figure 10: Repeater insertion for a large reset net.

6 Conclusion

We have presented an approach to repeater insertion and wire sizing that is based on the optimal Lillis/Ginneken dynamic programming framework. However, using state-of-the-art delay modeling techniques and an elegant hierarchical approach to traditional moment-matching techniques like Asymptotic Waveform Evaluation we have reasonably incorporated important transition time constraints into the repeater insertion framework. Furthermore, our comparison with SPICE shows that our algorithm produces solutations that are within 5% of the specified delay and transition-time constraints.

References

[1] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," Journal of Applied Physics, vol. 19, no. 1, 1948.

[2] P. Penfield and J. Rubinstein, "Signal delay in RC tree networks," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 202-211, July 1983.

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

[4] N. Menezes, R. Baldick, and L. T. Pileggi, "A sequential quadratic programming approach to concurrent gate and wire sizing," *IEEE Trans. Computer-Aided Design*,

[5] C.-P. Chen, Y.-P. Chen, and D.F. Wong, "Optimal wire-sizing formula under the Elmore delay model," *Proc. 33rd ACM/IEEE Design Automation Conferenc*e, pp. 487–490, 1994.

[6] J. Lillis. C.-K. Cheng, and T.-T. Lin, "Optimal and efficient buffer insertion and wire sizing," *Proc. Custom Integrated Circuits Conference*, pp. 259–262, May 1995.

[7] L. P. P. P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal Elmore Delay," *Proc. International Symposium onCircuits and Systems*, pp. 865-868, 1990.

[8] J. Cong and C.-K. Koh, "Simultaneous driver and wire sizing for performance and power optimization," *Proc. of the Intl. Conf. on Computer-Aided Design*, pp. 206–212, Nov. 1994.

[9] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, "Fidelity and near-optimality of Elmore-based routing constructions," *Proc. of the Intl. Conf. Computer Design*, pp. 81–84, Oct. 1993.

[10] P. Feldman and R.W. Freund, "Efficient linear circuit analysis by Pade approximation via the Lanczos process," *IEEE Trans. Computer-Aided Design.*, vol. 14, no. 5, pp. 639-649, May 1995.

[11] M. Sriram and S.-M. Kang, *Physical design of multichip modules*, Kluwer Academic Publishers, Boston, MA, 1994.

[12] F. Dartu, N. Menezes, J. Qian, and L.T. Pillage, "A gate-delay model for high-speed CMOS circuits," *Proc. 31st ACM/IEEE Design Automation Conferenc*e, pp. 576–580, June 1994.

[13] J. Qian, S. Pullela and L. T. Pillage, "Modeling the *effective capacitance* for the RC interconnect of CMOS gates," *IEEE Trans. Computer-Aided Design.*, vol. 13, no. 12, pp. 1526-1535, Dec. 1994.

[14] J. K. Ousterhout, "A switch-level timing verifier for digital MOS VLSI," *IEEE Trans. Computer-Aided Design*, vol. 4, no. 3, pp. 336-349, July 1985.

[15] 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*, Nov. 1989.

[16] S. Sapatnekar, "RC interconnect optimization under the Elmore delay model," *Proc. 31st Design Automation Conference*, pp. 387-391, June 1994.

[17] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. Computer-Aided Design*, vol. 9, no. 4, pp. 352-366, April 1990.