Q-Tree: A New Iterative Improvement Approach for Buffered Interconnect Optimization*

Andrew B. Kahng and Bao Liu†
CSE and ECE Departments, UC San Diego, La Jolla, CA 92093
abk@ucsd.edu,bliu@cs.ucsd.edu

Abstract

The "chicken-egg" dilemma between VLSI interconnect timing optimization and delay calculation suggests an iterative approach. We separate interconnect timing transformation as Hanan grafting and non-Hanan sliding, and reveal generally negligible contribution of non-Hanan sliding. We propose a greedy iterative interconnect timing optimization algorithm called Q-Tree. Our experimental results show that Q-Tree starting with Steiner minimum tree topologies achieves better timing performance than C-Tree [1], PER-Steiner [5] and BA-Tree [14] algorithms. Also, executing Q-Tree starting with BA-Tree or P-Tree [13] topologies can achieve better timing performance, especially, with shorter wires and fewer buffers. In general, Q-Tree can be applied to any interconnect tree for further timing performance improvement, with practical instance sizes and easily-extended functionality - e.g., with buffer station and routing obstacle avoidance consideration.

1 Introduction

The major objective of VLSI interconnect synthesis has shifted from area minimization to performance optimization. Design convergence - in particular timing closure - has become one of the most critical concerns of any design methodology. Logic synthesis must be revoked with no guarantees of improvement if physical design cannot meet timing assumptions from logic synthesis. Interconnect timing optimization has become increasingly critical in achieving timing closure.

Existing VLSI interconnect construction algorithms (Table 1) have various well-understood limitations in their fields of use. Traditional minimum-spanning tree (MST) (e.g., Prim) or Steiner minimum tree (SMT) (e.g., ER-Steiner [5]) interconnect topologies are no longer timing optimum in deep submicron (DSM) technologies due to non-negligible interconnect resistance. Shortest path trees (SPT) and arborescences (e.g., A-Tree [7]), or shallow-light tree[1] [3] (e.g., AHHK [2], BPRIM and BRBC [6]) are insensitive to electrical parameters (e.g., driver strength, sink load capacitance, required-arrival times) and thus give the same results over all technologies, pin loads, driver strengths, etc. Elmore-delay-based timing optimization heuristics (e.g., C-Tree [1], ERT, SERT, SERT-C [4], PER-Steiner [5] and Alphabetic tree [15]) do not guarantee timing performance: e.g., C-Tree solutions are limited in 5 empirical AHHK-over-SMT tree topologies; and PER-Steiner constructs shortest paths to pre-identified critical sinks before further improvements. Dynamic programming approaches (e.g., BA-Tree [14], MBA-Tree, RMP [8], P-Tree [13] and S-Tree [11]) can achieve optimum area or timing performance, and can extend to address such functionality as simultaneous buffering, use of buffer stations, and routing obstacle avoidance. However, dynamic programming approaches are only weakly scalable, e.g., to nets with up to 10 [8] or 15 [11] sinks even with pruning techniques that sacrifice optimality.

	SPT	MST	SMT
Min Area		Prim	ER-Steiner [5]
Linear Delay	A-Tree [7]	AHHK [2] BPRIM [6] BRBC [6]	
Elmore Delay		C-Tree [1] ERT [4]	SERT [4] PER-SERT-C [4] Steiner [5] Alpha [15]
w/ Sim. Buffering	BA-Tree [14]		P-Tree [13]
w/ Buffer Station + Routing Obstacle	MBA-Tree [8]		RMP [8] **Q-Tree** S-Tree [11]

Table 1: Interconnect routing tree construction algorithms categorized by topologies and objectives/functionalities.

In this paper, we propose a new iterative improvement

*This work was partially supported by Cadence Design Systems, Inc., the MARCO Gigascale Silicon Research Center and NSF Grant CCR-9988331.

†Incentia Design Systems, Inc., Santa Clara, CA 95054.

[1] A shallow-light tree achieves small radius or small maximum source-to-sink pathlength in the tree (i.e., "shallow") and small cost(i.e., "light") at the same time.

approach for buffered interconnect timing optimization. We present our problem formulation in Section 2. Section 3 analyzes iterative interconnect timing optimization approaches, which we separate as Hanan grafting and non-Hanan sliding. A greedy iterative interconnect timing optimization algorithm Q-Tree is developed in Section 4. We present our experimental results in Section 5 and conclude in Section 6.

2 Notations and Problem Formulation

We adopt the following notations in this paper.

- r = per unit length wire resistance,
- c = per unit length wire capacitance,
- R_s = driver output resistance of source s of interconnect T,
- R_b = buffer output resistance,
- C_b = buffer input capacitance,
- D_b = internal buffer delay,
- $C_t(i)$ = capacitance of sink $i \in K$,
- $q(v)$ = required-arrival time of node v,
- $R_U(u)$ = total resistance of the path up to the nearest upstream buffer or source, including the buffer or source output resistance,
- $C_T(v)$ = total capacitance of the subtree rooted at node v down to the nearest downstream buffer or sinks (i.e., no buffer is properly included in the subtree), including the sink or buffer input capacitance,
- $l(u, v)$ = edge length between nodes u and v,
- $p_T(u, v)$ = path in tree T between nodes u and v.

A DSM VLSI interconnect can be represented as a RC tree with segment resistances and capacitances, resistive driver, and capacitive loads. The presence of buffers B separates an interconnect into stages. Elmore delay [9] from the source s to a sink k is given by:

$$d_E(k) = \sum_{i \in \{k\} \cup (B \cap p_T(s,k))} d_E^{stage}(i) \quad (1)$$

where path delay in a stage driven by $b \in \{s\} \cup B$ is given by:

$$d_E^{stage}(i) = \sum_{(u,v) \in p_T(b,i)} rl(u,v)(0.5cl(u,v) + C_T(v)) + R_s C_T(b) + D_b. \quad (2)$$

The Interconnect Timing Optimization (ITO) problem is as follows: Given

1. unit interconnect resistance r,
2. unit interconnect capacitance c,
3. source s with driving resistance R_s,
4. sink set K with capacitance $C_t(i)$ and required-arrival time $q(i)$ $\forall i \in K$,
5. buffer library L with input capacitance $C_b(t)$, output resistance $R_b(t)$, internal delay $D_b(t)$ (and location $(x, y)(t)$ if t is a buffer station) $\forall t \in L$,

construct a tree $T(B \cup \{s\} \cup K)$ where every $b \in B$ is an instance of some buffer type $t \in L$, such that

1. source required-arrival time $q(s)$ is maximized,
2. $q(s) + d_E(i) \leq q(i)$, $\forall i \in K$.

By setting unit interconnect resistance $r = 0$ with an empty buffer library $L = \emptyset$, we see that RSMT reduces to ITO, which shows that ITO is NP-hard.

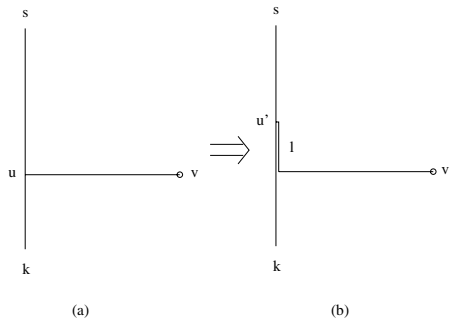

Figure 1: Elmore delay $d_E(k)$ could be decreased by sliding a Steiner point u upstream by distance l to u' .

3 Analysis

In this section, we analyze two iterative interconnect timing optimization primitives called *Hanan grafting* and *non-Hanan sliding*. Our analyses reveal limited contribution of non-Hanan sliding, and form the foundation of the Q-Tree algorithm in the next section.

3.1 Sliding

A timing optimum interconnect tree may not be a RSMT on the Hanan grid,[2] since Elmore delay to a sink can be decreased by *non-Hanan sliding* [10, 15].

Definition 1 *Non-Hanan* sliding *performs interconnect tree transformation by relocating a Steiner node towards its parent node, i.e., to a non-Hanan location with introduction of coincident interconnect segments.*

For a Steiner node u slid upstream by distance l to location u' (Figure 1), [3] the change of Elmore delay to sink k per unit extra wirelength (= sliding distance) is given by:

$$\Delta d_E^{sl}(k) = (R_U(u) - rl)cl - rl(C_T(v) + cl(u,v)) = rcl(l^*(u) - l) \quad (3)$$

[2]The Hanan grid is formed by passing horizontal and vertical lines through every terminal $i \in \{s\} \cup K$.

[3]We consider a binary tree for simplicity. A general tree can be transformed to a binary tree by introducing additional Steiner nodes and zero-length edges.

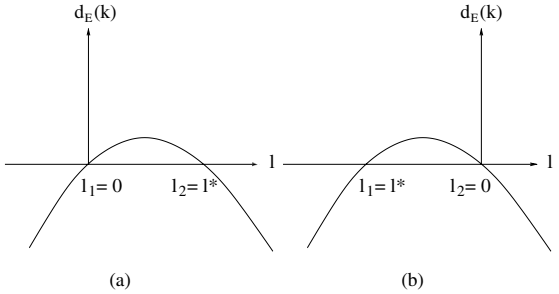

Figure 2: Elmore delay $d_E(k)$ is a concave quadratic function of sliding distance l with two roots: 0 and an either (a) positive or (b) negative $l^*(u)$.

where

$$l^*(u) \stackrel{def}{=} \frac{R_U(u)}{r} - \frac{C_T(v)}{c} - l(u,v).$$

Observation 1 *Elmore delay to sink k is decreased (i.e., $\Delta d_E^{sl}(k) \leq 0$), by sliding a Steiner node u upstream by distance l if and only if $l^*(u) \leq l$.*

Observation 2 *The smaller is $l^*(u)$, the larger is the Elmore delay decrease per unit extra wirelength.*

Note that Elmore delay is a *concave* function of sliding distance l (Figure 2). The following observations hold.

Observation 3 *Minimum Elmore delay to the sink k is achieved by sliding the Steiner node u to an extreme location (i.e., its current location or the location of its parent node), which is a Hanan location if the original routing is Hanan.*

Observation 4 *Maximum required-arrival time at source of a Hanan routing is achieved by sliding the Steiner node u to a Hanan location, unless the extra wirelength makes a different (off-path) sink critical.*

3.2 Grafting

Complementary to non-Hanan sliding, which embeds the same tree topology beyond the Hanan grid, *Hanan grafting* embeds a different tree topology on the Hanan grid. Hanan grafting extends the basic edge replacement operation proposed in [5] by allowing possible buffer insertion.

Definition 2 *Hanan* grafting *performs tree transformation by possibly buffered edge replacement on the Hanan grid.*[4]

Consider a grafting which replaces a tree edge (u,v), $u = parent(v)$, by a non-tree edge (u',v'). The change of Elmore delay $\Delta d_E^{gr}(k)$ to sink k can be calculated for each of the following cases (Figure 3): (a) $v' \notin p_T(s,k)$, (b) $v' \in p_T(s,k)$, $u' \notin p_T(s,k)$, and (c) $v' \in p_T(s,k)$, $u' \in p_T(s,k)$, each with possible buffer insertion. E.g., in

[4]Depending on different scenarios, buffers can be inserted at the head of the edge, at the head of edge segments, at buffer stations, optimally, etc.

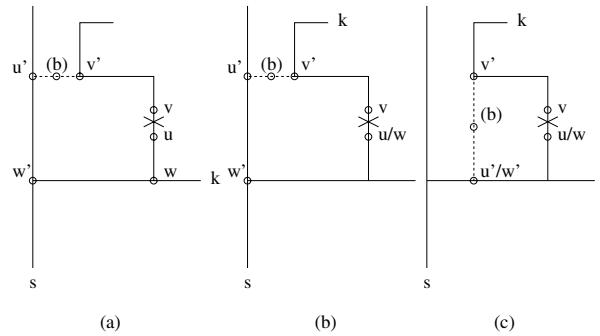

Figure 3: Grafting a routing tree by replacing a tree edge (u,v) by a non-tree edge (u',v') with a possible buffer b inserted: (a) $v' \notin p_T(s,k)$, (b) $v' \in p_T(s,k), u' \notin p_T(s,k)$, (c) $v' \in p_T(s,k), u' \in p_T(s,k)$.

case that $v' \notin p_T(s,k)$, without buffer insertion, the Elmore delay change is given by:

$$\Delta d_E^{gr}(k) = R_U(w')c(l_2 - l_1) - rl(w,w')C_T(v) \quad (4)$$

or, with a buffer b inserted,

$$\Delta d_E^{gr}(k) = R_U(w')(c*l_{2b} - c*l_1 - C_T(v) + C_b(b)) - rl(w,w')C_T(v) \quad (5)$$

where $l_1 = l(u,v)$, $l_2 = l(u',v')$, $l_{2a} = d(v',b)$, $l_{2b} = d(b,u')$, if a buffer b is inserted on (u',v'), $w = lca(k,v)$ (resp. $w' = lca(k,v')$) is the least common ancestor of k and v (resp. k and v').

We observe that (i) the optimum sliding is Hanan when starting with a Hanan routing, unless the extra wirelength makes a different sink critical; and (ii) Hanan sliding is a special case of Hanan grafting, i.e., case (a) without buffer insertion. The following observation holds.

Observation 5 *Without introduction of extra wirelength which makes a different sink critical, a Hanan grafting achieves the largest source required-arrival time $q(s)$ decrease per unit extra wirelength (i.e., $\Delta d_E^{gr}(k) \leq \Delta d_E^{sl}(k) \leq 0$, where k is the critical sink); otherwise non-Hanan sliding makes further timing performance improvement.*

Observation 5 forms the basis of our proposed greedy iterative timing optimization algorithm - Q-Tree.

4 Greedy Iterative Optimization

We propose a greedy iterative timing optimization algorithm Q-Tree based on the above efficiency analysis. Q-Tree chooses the most efficient interconnect tree transformation with the largest source required-arrival time increase per unit extra wirelength (Algorithm 1). Hanan grafting is preferred over non-Hanan sliding whenever possible due to Observation 5.

Observing that in many cases shorter wires imply smaller delay, we classify grafting into three categories depending on whether the wirelength is (i) decreased, (ii) unchanged or (iii) increased. Each category is sorted respectively by (i) amortized critical sink delay decrease $K_A = -\Delta d_E(k) - (l_2 - l_1)avg(\frac{-\Delta d_E(k)}{l_2 - l_1})$, where $-\Delta d_E(k)$ is the actual critical sink delay decrease, $avg(\frac{-\Delta d_E(k)}{l_1 - l_2})$ is the average critical sink delay decrease per unit extra wirelength, and $(l_2 - l_1)avg(\frac{-\Delta d_E(k)}{l_1 - l_2})$ is the expected future critical sink delay decrease due to current wirelength saving $l_1 - l_2$, (ii) critical sink delay decrease $K_B = -\Delta d_E(k)$, and (iii) critical sink delay decrease per unit extra wirelength $K_C = -\frac{\Delta d_E(k)}{l_2 - l_1}$. In choosing the most efficient grafting, decreased wirelength is preferred over unchanged wirelength, which is preferred over increased wirelength. The chosen grafting is then verified by updating all sink Elmore delays. If no other sink becomes more timing critical, the chosen grafting is committed by updating the routing tree. Otherwise the next efficient grafting candidate is verified until finding a committable grafting or reaching the end of the lists (Algorithm 2).

The most efficient sliding is at a Steiner node u with the smallest $l^*(u) \leq l(u, parent(u))$ (Equation 3). Node u is slid upstream by distance l to maximize the source required-arrival time $q(s) = Min_{i \in K}\{q(i) - d_E(s,i)\}$, where Elmore delay $d_E(i)$ is quadratic in l if $i \in T(u)$ (Equation 3), or linear in l if $i \notin T(u)$.[5]

The overall runtime for Q-Tree is $O(Mn^3|L|\log(n^3|L|))$, where M is the number of graftings tried in a Q-Tree run, n is the tree size, and $|L|$ is the buffer library size.

5 Experiments

In the following experiments we collect data over 5, 10, 15, 20, 50 and 100 terminals. For each terminal number, 100 sets of terminal locations are randomly generated in a $10,000\mu m \times 10,000\mu m$ square, over which interconnects are constructed based on $180nm$, $130nm$, $100nm$ and $70nm$ ITRS parameters and the Berkeley Predictive Technology Model (BPTM) beta version[12] (Table 2). Results in terms of average source required-arrival time, wirelength, buffer number and runtime are presented with identical driver, buffer and sink size ($R_s = R_b, C_b = C_t$) and identical sink required-arrival times.[6] The runtimes are measured on a $1.4GHz$ Intel Xeon $i686$ with $1GB$ memory, excluding the time for SMT construction by the ER-Steiner heuristic.

[5]The optimum sliding distance l can be found by calculating the intersections of these quadratic or linear functions, which takes $O(n^2)$ runtime. We adopt a binary search approach (Algorithm 3), which takes $O(n \log l)$ runtime with negligible suboptimality as observed in our experiments.

[6]Similar results are obtained for (i) instances with uniformly distributed sink capacitances in $(0, C_b)$ and (ii) instances with sink required-arrival times uniformly distributed in $(0, R_b * \sum_{i \in K} C_t(i))$.

Algorithm 1: Q-Tree
Input: r, c, R_s, $C_t(i)$ and $q(i)$ $\forall i \in K$, $C_b(t)$ and $R_b(t)$ $\forall t \in B_l$.
Output: Routing tree T with maximized source required-arrival time $q(s)$.

```
Construct, if not given, an initial tree T
Compute Elmore delay d_E(t) for each sink t
For each critical sink k with minimum slack q(k) − d_E(k)
    If grafting returns timing improved routing tree T
        Continue
    Else if sliding returns timing improved routing tree T
        Continue
    Else
        Return T
```

Algorithm 2: Grafting
Input: Routing tree T, critical sink k, buffer library B_l.
Output: Timing improved or unchanged routing tree T.

```
For each pair of nodes v' and u' ∉ T_v'
    For each buffer type t ∈ L ∪ ∅
        For each node v ∈ T_w, w = lca(v', u') and v' ∈ T_v
            Compute Δd_E(k)
            If l_2 < l_1
                Push grafting (u', v', v, K_A) into queue Q_A
            Else if l_2 = l_1
                Push grafting (u', v', v, K_B) into queue Q_B
            Else if l_2 > l_1
                Push grafting (u', v', v, K_C) into queue Q_C
            Update avg(Δd_E(k) / (l_2 − l_1)) for amortization
Sort Q_A with key K_A in increasing order
Sort Q_B with key K_B in increasing order
Sort Q_C with key K_C in increasing order
While true
    If Q_A ≠ ∅
        Pop up a grafting from Q_A
    Else if Q_B ≠ ∅
        Pop up a grafting from Q_B
    Else if Q_C ≠ ∅
        Pop up a grafting from Q_C
    Else
        Return unchanged routing tree T
    Compute q(s)
    If q(s) increases
        Return timing improved routing tree T
```

Algorithm 3: Sliding
Input: Routing tree with critical sink k.
Output: Timing improved or unchanged routing tree T.

```
For each Steiner node u, k ∈ T_u
    Compute l*(u)
Find node u* with minimum l*(u*) < l(u*, parent(u*))
Find sliding distance l
Update Elmore delay d_E(t) for each sink t ∈ T
If q(s) increases
    Return improved routing tree T
Else
    Return unchanged routing tree T
```

We first compare three (unbuffered) interconnect topology optimizations: Q-Tree starting with ER-Steiner, C-Tree and PER-Steiner (Table 3). We observe that significant interconnect timing performance improvement over Steiner minimum trees can be achieved by interconnect topology optimization at the expense of moderate wirelength increase. This improvement grows with technology advancement and instance size increase. Further, Q-Tree topology

	180nm	130nm	100nm	70nm
$r(\Omega/\mu m)$	0.040	0.098	0.186	0.391
$c(fF/\mu m)$	0.232	0.212	0.191	0.155
$R_b(\Omega)$	139.434	186.826	226.800	250.063
$C_b(fF)$	63.358	33.652	17.370	6.869
$D_b(ps)$	23.348	20.830	19.949	15.040
$R_bC_b + D_b(ps)$	32.182	27.117	23.888	16.757

Table 2: ITRS and BPTM parameters for $180nm$, $130nm$, $100nm$ and $70nm$ technologies.

		ER|Q-Tree			C-Tree			PER-Steiner		
	n	$\|q(s)\|$	$l(T)$	CPU	$\|q(s)\|$	$l(T)$	CPU	$\|q(s)\|$	$l(T)$	CPU
180	10	0.780	1.346	0.004	0.829	1.178	0.014	0.989	1.155	0.000
	20	0.657	1.297	0.107	0.722	1.200	0.066	0.957	1.136	0.001
	50	0.544	1.254	4.779	0.639	1.255	0.941	0.933	1.069	0.005
	100	0.507	1.134	52.415	0.623	1.214	9.556	0.841	1.045	0.020
130	10	0.687	1.399	0.005	0.704	1.257	0.012	0.981	1.104	0.000
	20	0.569	1.449	0.109	0.598	1.281	0.065	0.945	1.174	0.001
	50	0.459	1.367	4.729	0.506	1.266	0.941	0.878	1.070	0.005
	100	0.395	1.206	53.894	0.482	1.231	9.592	0.781	1.051	0.022
100	10	0.636	1.595	0.006	0.630	1.250	0.012	0.962	1.207	0.000
	20	0.522	1.446	0.119	0.527	1.305	0.064	0.931	1.110	0.001
	50	0.410	1.384	4.207	0.432	1.287	0.943	0.829	1.078	0.006
	100	0.338	1.235	51.752	0.412	1.231	9.610	0.662	1.056	0.021
70	10	0.558	1.617	0.004	0.560	1.345	0.014	0.900	1.097	0.000
	20	0.450	1.597	0.134	0.457	1.305	0.066	0.839	1.098	0.001
	50	0.337	1.483	4.810	0.363	1.359	0.948	0.585	1.058	0.005
	100	0.280	1.254	54.801	0.328	1.262	9.627	0.385	1.028	0.023

Table 3: Average source required-arrival time $q(s)$, wirelength $l(T)$ (normalized to ER-Steiner [5]), and runtime of *unbuffered* (a) Q-Tree starting with ER-Steiner, (b) C-Tree and (c) PER-Steiner over 100 randomly generated terminal sets with identical sink capacitances and required-arrival times under $180nm$, $130nm$, $100nm$ and $70nm$ technology, respectively.

optimization starting with ER-Steiner topologies achieves better timing performance with longer wires than C-Tree and PER-Steiner topology optimizations.

We observe indistinguishable timing performance improvement by Q-Tree topology optimization with or without non-Hanan sliding. Since introduction of buffer insertion would further decrease the contribution of sliding, i.e., as an alternative of critical sink isolation, we have: Non-Hanan sliding contributes generally negligible timing performance improvement.

We apply Q-Tree to ER-Steiner, BA-Tree and P-Tree interconnects.[7] For each interconnect, Q-Tree is first applied without any bound of wirelength or buffer number to achieve the best possible timing performance. Bound of wirelength and buffer number is then applied to achieve a "dominant" Q-Tree solution (i.e., with shorter wires, fewer buffers and better timing performance). We observe that (Table 4):

1. Q-Tree starting with ER-Steiner in average achieves better timing performance than BA-Tree with longer wires and more buffers, and worse timing performance than P-Tree with shorter wires and fewer buffers.
2. Q-Tree starting with ER-Steiner can achieve dominant solutions over BA-Tree.
3. Q-Tree starting with BA-Tree can achieve better timing performance, especially, dominant solutions over BA-Tree.[8]
4. Q-Tree starting with P-Tree can achieve better timing performance, especially, dominant solutions over P-Tree.[9]

6 Conclusion

We propose iterative interconnect timing optimization as a solution to the "chicken-egg" dilemma between VLSI interconnect timing optimization and delay calculation. We separate iterative optimization approaches as Hanan *grafting* and non-Hanan *sliding*. Our analyses reveal limited contribution of non-Hanan sliding on timing performance improvement. We also propose a greedy iterative interconnect timing optimization algorithm Q-Tree. Our experimental results show that Q-Tree starting with ER-Steiner (resp. BA-Tree or P-Tree) can achieve better timing performance, especially, with shorter wires and fewer buffers compared to BA-Tree (resp. BA-Tree or P-Tree). In general, Q-Tree can be applied to any interconnect tree for further timing performance improvement, with practical instance sizes and easily-extended functionality, e.g., with buffer station and routing obstacle consideration. Q-Tree provides a simple and powerful VLSI interconnect timing optimization approach.

The authors thank Professor John Lillis, Milos Hrkic, and Dr. Manjit Borah for their help with the P-Tree and PER-Steiner codes, and Dr. Ion Mandoiu for useful comments on the experimental setup.

References

[1] C. J. Alpert, M. Hrkic, J. Hu, A. B. Kahng, J. Lillis, B. Liu, S. T. Quay, S. S. Sapatnekar, A. J. Sullivan, and P. Villarrubia. Buffered Steiner trees for difficult instances. In *ACM/SIGDA International Symposium on Physical Design*, pages 4–9, 2001.

[2] C. J. Alpert, T. C. Hu, J. H. Huang, A. B. Kahng, and D. Karger. Prim-Dijkstra tradeoffs for improved performance-driven routing tree design. *IEEE Transactions on Computer-Aided Design*, 14(7), 1995.

[3] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensitive analysis of communication protocols. In *Proc. ACM Symp. on Principles of Distributed Computing*, pages 177–187, 1990.

[7] C-Tree solutions on randomly generated instances are not available due to instability of C-Tree code. P-Tree results on 50 sink instances are not available due to weak scalability.

[8] Q-Tree dominates BA-Tree by better construction, i.e., in merging branches in different quadrants.

[9] Q-Tree dominance over P-Tree is possible due to P-Tree suboptimality, i.e., buffers can only be inserted at a branching point.

n		ER|Q	ER|Q^*	BA	BA|Q	BA|Q^*	P	P|Q	P|Q^*	ER|Q	ER|Q^*	BA	BA|Q	BA|Q^*	P	P|Q	P|Q^*
		180								100							
5	$\|q(s)\|$	0.638	0.889	0.912	0.165	0.828	0.487	0.204	0.242	0.552	0.808	0.827	0.167	0.750	0.328	0.174	0.203
	$l(T)$	1.255	1.130	1.141	1.087	1.035	1.363	1.359	1.326	1.311	1.242	1.141	1.168	1.035	1.341	1.326	1.326
	$\|B\|$	5.190	0.410	0.810	3.240	0.810	6.260	6.320	5.870	5.700	0.250	0.490	3.020	0.490	7.610	7.090	6.830
	CPU	0.010	0.000	0.000	0.010	0.000	0.031	0.063	0.044	0.016	0.000	0.000	0.010	0.000	0.026	0.072	0.064
10	$\|q(s)\|$	0.433	0.612	0.844	0.511	0.680	0.289	0.120	0.128	0.387	0.605	0.714	0.066	0.491	0.167	0.121	0.122
	$l(T)$	1.578	1.171	1.293	1.318	1.281	1.775	1.766	1.751	1.653	1.261	1.295	1.502	1.128	1.625	1.625	1.614
	$\|B\|$	12.830	1.680	2.630	4.780	2.630	13.890	13.630	12.740	13.510	0.680	0.850	5.110	0.850	18.200	16.550	16.340
	CPU	0.402	0.017	0.000	0.045	0.010	6.283	1.012	0.885	0.474	0.017	0.000	0.095	0.012	4.035	1.009	1.008
15	$\|q(s)\|$	0.351	0.633	0.703	0.419	0.560	0.225	0.095	0.114	0.288	0.566	0.622	0.174	0.400	0.124	0.085	0.085
	$l(T)$	1.665	1.222	1.268	1.268	1.268	1.881	1.879	1.866	1.759	1.292	1.267	1.506	1.138	1.667	1.660	1.657
	$\|B\|$	18.050	1.720	4.090	6.030	4.090	20.540	16.870	16.220	20.620	0.730	2.230	6.130	2.230	24.370	20.370	20.080
	CPU	1.653	0.080	0.000	0.191	0.018	201.339	1.018	1.002	2.310	0.053	0.000	0.375	0.060	98.241	1.028	1.011
50	$\|q(s)\|$	0.184	0.392	0.521	0.348	0.400	-	-	-	0.154	0.314	0.477	0.291	0.323	-	-	-
	$l(T)$	1.742	1.306	1.433	1.433	1.433	-	-	-	1.762	1.349	1.413	1.413	1.413	-	-	-
	$\|B\|$	41.680	5.590	8.440	10.240	8.440	-	-	-	38.540	5.010	8.070	10.030	8.070	-	-	-
	CPU	76.404	9.634	0.012	7.359	0.460	-	-	-	77.438	9.996	0.012	7.522	0.504	-	-	-
		130								70							
5	$\|q(s)\|$	0.577	0.846	0.859	0.176	0.780	0.387	0.204	0.204	0.525	0.763	0.793	0.159	0.719	0.268	0.164	0.191
	$l(T)$	1.317	1.242	1.141	1.168	1.035	1.383	1.376	1.365	1.341	1.242	1.141	1.168	1.035	1.354	1.316	1.318
	$\|B\|$	5.440	0.240	0.420	2.890	0.420	7.530	7.290	6.780	5.490	0.260	0.400	3.040	0.400	8.280	7.780	7.400
	CPU	0.015	0.000	0.000	0.010	0.000	0.020	0.071	0.056	0.023	0.000	0.000	0.010	0.000	0.022	0.079	0.067
10	$\|q(s)\|$	0.391	0.560	0.762	0.271	0.527	0.211	0.118	0.117	0.376	0.564	0.654	0.012	0.491	0.122	0.070	0.070
	$l(T)$	1.620	1.213	1.295	1.358	1.230	1.685	1.682	1.672	1.686	1.345	1.290	1.482	1.093	1.553	1.553	1.545
	$\|B\|$	13.790	1.330	1.860	4.640	1.860	16.950	15.420	15.190	13.420	0.270	0.420	5.460	0.420	19.520	18.030	17.830
	CPU	0.488	0.019	0.000	0.059	0.020	4.654	1.023	1.003	0.465	0.021	0.000	0.120	0.013	3.298	1.012	1.009
15	$\|q(s)\|$	0.309	0.576	0.676	0.322	0.459	0.162	0.075	0.084	0.274	0.612	0.579	0.007	0.348	0.086	0.063	0.064
	$l(T)$	1.778	1.228	1.258	1.388	1.220	1.811	1.811	1.792	1.791	1.353	1.234	1.493	1.049	1.614	1.614	1.611
	$\|B\|$	20.580	1.390	3.030	5.160	3.030	22.990	19.110	18.360	20.830	0.390	0.660	5.850	0.660	28.370	22.980	22.710
	CPU	2.356	0.065	0.000	0.214	0.026	133.807	1.029	1.009	2.518	0.053	0.000	0.419	0.053	79.318	1.038	1.013
50	$\|q(s)\|$	0.174	0.317	0.464	0.309	0.340	-	-	-	0.162	0.285	0.453	0.275	0.361	-	-	-
	$l(T)$	1.779	1.401	1.443	1.443	1.443	-	-	-	1.796	1.373	1.410	1.410	1.410	-	-	-
	$\|B\|$	37.390	6.120	9.280	11.250	9.280	-	-	-	38.180	3.750	6.450	8.610	6.450	-	-	-
	CPU	68.412	10.437	0.010	7.912	0.499	-	-	-	80.272	9.275	0.015	7.314	0.466	-	-	-

Table 4: Average source required-arrival time $q(s)$, wirelength $l(T)$ (normalized to ER-Steiner [5]), buffer number $|B|$ and runtime (in seconds) achieved by (a) Q-Tree starting with ER-Steiner (ER|Q), (b) Q-Tree starting with ER-Steiner with $l(T)$ and $|B|$ bounded by $1.1\times$ BA-Tree results (ER|Q^*), (c) BA-Tree (BA), (d) Q-Tree starting with BA-Tree (BA|Q), (e) Q-Tree starting with BA-Tree with $l(T)$ and $|B|$ bounded by $1.0\times$ BA-Tree results (BA|Q^*), (f) P-Tree (P), (g) Q-Tree starting with P-Tree (P|Q), (h) Q-Tree starting with P-Tree with $l(T)$ and $|B|$ bounded by $1.1\times$ P-Tree results (P|Q^*), over 100 randomly generated sets of terminals with identical sink capacitances and required-arrival times under $180nm$, $130nm$, $100nm$ and $70nm$ technology, respectively.

[4] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins. Fidelity and near-optimality of Elmore-based routing constructions. pages 81–84, 1993.

[5] M. Borah, R. M. Owens, and M. J. Irwin. A fast and simple Steiner routing heuristic. *Discrete Applied Mathematics*, 90:51–67, 1999.

[6] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong. Provably good performance-driven global routing. *IEEE Transactions on Computer-Aided Design*, 11(6):739–752, 1992.

[7] J. Cong, K. S. Leung, and D. Zhou. Performance-driven interconnect design based on distributed RC delay mode. In *ACM/IEEE Design Automation Conference*, pages 606–611, 1993.

[8] J. Cong and X. Yuan. Routing tree construction under fixed buffer locations. In *ACM/IEEE Design Automation Conference*, pages 379–384, 2000.

[9] W. C. Elmore. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, 19:55–63, 1948.

[10] H. Hou, J. Hu, and S. S. Sapatnekar. Non-Hanan routing. *IEEE Transactions on Computer-Aided Design*, 18(4):436–444, 1999.

[11] M. Hrkic and J. Lillis. Buffer tree synthesis with consideration of temporal locality, sink polarity requirements, solution cost and blockages. In *ACM/SIGDA International Symposium on Physical Design*, pages 362–367, 2002.

[12] http://www device.eecs.berkeley.edu/∼ptm/.

[13] J. Lillis, C.-K. Cheng, T.-T. Y. Lin, and C.-Y. Ho. New performance driven routing techniques with explicit area/delay tradeoff and simultaneous wire sizing. In *ACM/IEEE Design Automation Conference*, pages 395–400, 1996.

[14] T. Okamoto and J. Cong. Buffered Steiner tree construction with wire sizing for interconnect layout optimization. In *Proceedings IEEE-ACM International Conference on Computer-Aided Design*, pages 44–49, 1996.

[15] A. Vittal and M. Marek-Sadowska. Minimal delay interconnect design using alphabetic trees. In *ACM/IEEE Design Automation Conference*, pages 392–396, 1994.