Short Papers

An Effective Algorithm for Buffer Insertion in General Circuits Based on Network Flow

Ruiming Chen and Hai Zhou

***Abstract*—The problem of buffer insertion in a single net has been the focus of most previous research works. However, effective algorithms for buffer insertion in whole circuits are generally needed. In this paper, we relate the timing-constrained minimal buffer insertion problem to the convex cost-flow dual problem and propose an algorithm based on the *convex cost-flow* theory to solve it in combinational circuits. Experimental results demonstrate that our approach is effective. On the average, for the cases where buffering locations are not specified, our approach achieves a 46% reduction on the total buffer area in comparison to a traditional approach; for the cases where buffering locations are specified, our approach achieves a 52% reduction.**

***Index Terms*—Buffer insertion, network flow, timing optimization.**

I. Introduction

Buffer insertion is widely used to reduce interconnect delays [1], [2], [7], [11], [14], [15]. Van Ginneken [14] proposed a dynamic programming method to buffering in distributed resistance–capacitance tree networks for minimal Elmore delay. Lillis *et al.* [7] extended van Ginneken's algorithm to minimize the power consumption with more general delay models. However, most of the existing approaches considered the buffer insertion problem only on a single net. Recent projections of historical scaling trends by Saxena *et al.* [10] predict synthesis blocks to have 70% of their cell count dedicated to interconnect buffers within a few process generations. The power consumption of these buffers becomes prominent, so it is necessary to reduce buffers without sacrificing the performance of a circuit.

If the given timing constraint is greater than the minimal delay that can be achieved by buffering, assignment of various timing budgets on the critical paths would give multiple buffering solutions, among which we need to select the one with the minimal cost. This problem also occurs on buffering the less critical paths even when the timing constraint is tight. Different from the buffering on a single net where the required times at sinks are known, the buffering on a circuit needs to do the timing budgeting, i.e., assign required time at sinks of nets. Thus, the buffering in a circuit involves two related steps: 1) budgeting and 2) single net buffering. Net-based buffering approaches do not have a timing budgeting step and consider the buffering problem in a local view. As will be demonstrated in our experiments, a net-based buffering approach obtains a significantly suboptimal solution. Liu *et al.* [9] presented a Lagrangian relaxation-based algorithm to solve the buffer insertion problem in large networks. It was extended to consider multiple buffer types and feasible buffer locations in [8].

Our experiments show that the results obtained using Lagrangian relaxation-based techniques are significantly suboptimal.

Manuscript received January 23, 2006; revised May 15, 2006, October 14, 2006, and January 2, 2007. This work was supported by the National Science Foundation under Grant CCR-0238484. This paper was recommended by Associate Editor D. Z. Pan.

The authors are with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208 USA (e-mail: rui-chen@northwestern.edu).

Digital Object Identifier 10.1109/TCAD.2007.906481

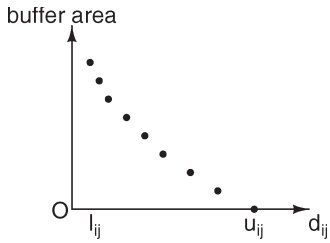

Fig. 1. Area of buffers as a function of the wire delays. l_{ij} and u_{ij} are the bounds of the delay d_{ij}.

Sze *et al.* [13] proposed a path-based buffering algorithm. The basic idea is to partition the whole circuit into a set of trees, treat the gates as specific buffers, and, for each tree, use the van Ginneken's algorithm to compute the solution. However, its performance when compared with the Lagrangian relaxation-based technique [8] is not reported.

The input to our problem is a placed and routed netlist of modules. Our objective is to insert buffers into wires in the general combinational circuit such that the timing constraint is met and the area of buffers is minimized. We relate this problem to the network-flow problem and present an effective algorithm based on the network-flow theory. Experimental results show that the solutions from our algorithm have much less cost than those from [8] and [9] and a net-based buffering approach.

The delay change of one component affects the delays of many other components. For example, the delay d_{ij} of wire (i, j) in a multipin net depends on not only the buffering on (i, j) but also the buffering of the fan-outs of (i, j). Without this kind of interaction, the buffering problem can be formulated as a network-flow problem, whereas with the interactions, the objective function is no longer separable, and thus, the problem becomes much more difficult to solve.

II. Network-Flow-Based Buffering

The buffer insertion problem can be viewed as a wire substitution problem. As shown in Fig. 1, a wire has different delays for different areas of buffers. The objective is to select a point in the configuration of each wire such that the total area of buffers is minimized while the timing constraints are satisfied. We propose an iterative approach: in each iteration, we find the best locations for buffers (i.e., a point on the configuration of the wire) and then replace the current wire by the wire with its configuration on that point. Thus, during the buffer insertion, the existing buffers actually move if those buffers are not in their best positions of the current iteration.

A. Theory for a Simplified Separable Model

In this section, for simplicity, we assume that there is only one type of buffer. We consider a simplified buffer insertion problem: the delay of a module is independent of the load of the module, and all the nets have only two pins. In this case, the delay of a wire does not depend on the buffering of its fan-outs, and the buffering of a wire changes only its own delay. Therefore, the number of buffers on a wire can be represented as a function of the delay of the wire. Let $K_{ij} = \mathcal{F}_{ij}(d_{ij})$, where $\mathcal{F}_{ij}$ is the function that gives the number of buffers K_{ij} for a given delay d_{ij}. In this paper, we only consider the buffer insertion in combinational circuits. Two dummy nodes s and t are introduced into

the graph representing the circuit: s is connected to all the primary inputs, and t is connected from all the primary outputs. We introduce one edge from t to s with weight $-REQ$, where REQ is the timing constraint. This new graph $G(V, E)$ is called a *constraint graph*.

As shown in Fig. 1, on the same wire, with more and more buffers inserted, the amount of delay reduction by inserting a buffer becomes smaller and smaller and will finally become nonpositive when the minimal delay is reached. Such a property of decreasing delay reduction with the increasing number of buffers is similar to the concept of convexity. For a complicated buffer library, the curve may not be strictly convex, but the general trend is convex. We connect each node to its nearest nodes, as shown in Fig. 1, to obtain a piecewise linear convex function $\mathcal{F}_{ij}$. For any edge (i, j), let l_{ij} be the minimal delay of edge (i, j) that can be achieved by buffering, and let u_{ij} be the delay without buffers. Let $l_{ts} = u_{ts} = -REQ$, and let $\mathcal{F}_{ij}(d_{ij})$ be equal to ∞ when $d_{ij} < l_{ij} \vee d_{ij} > u_{ij}$. It is obvious that if there is an optimal solution for the buffering problem, then $l_{ij} \leq d_{ij} \leq u_{ij}$. The timing-constrained buffering problem can be formulated as follows, given a constraint graph $G(V, E)$:

$$\text{Minimize} \sum_{(i,j)\in E} \mathcal{F}_{ij}(d_{ij})$$
$$\text{s.t.} \quad t_j \geq t_i + d_{ij} \qquad \forall (i,j) \in E. \tag{P1}$$

The constraint graph excluding the edge from t to s is a directed acrylic graph, $\mathcal{F}_{ij}$ has only nonnegative integer values, and d_{ij}'s can only be discrete values. To the best of our knowledge, currently, no algorithm can efficiently and optimally solve this problem.

We define $\mathcal{C}_{ij}(x) = \min_{x' \leq x} \mathcal{F}_{ij}(x')$; then, the following problem formulation can be proved to be equivalent to (P1), given a constraint graph $G(V, E)$:

$$\text{Minimize} \sum_{(i,j)\in E} \mathcal{C}_{ij}(d_{ij})$$
$$\text{s.t.} \quad t_j = t_i + d_{ij} \qquad \forall (i,j) \in E. \tag{P2}$$

The Karush–Kuhn–Tucker optimality conditions are obtained [6] if $\exists \mathbf{x} \in \mathcal{R}^E$ such that

$$t_j = t_i + d_{ij} \qquad \forall (i,j) \in E \tag{1}$$
$$-\mathcal{C}^+_{ij}(d_{ij}) \leq x_{ij} \leq -\mathcal{C}^-_{ij}(d_{ij}) \qquad \forall (i,j) \in E \tag{2}$$
$$\sum_{(i,j)\in E} x_{ij} - \sum_{(j,i)\in E} x_{ji} = 0 \qquad \forall i \in V \tag{3}$$

where $\mathcal{C}^-_{ij}(x)$ and $\mathcal{C}^+_{ij}(x)$ represent the left derivative and the right derivative of function $\mathcal{C}_{ij}$ at point x, respectively. Now, we briefly explain these conditions. Suppose a cut C splits the vertices into two sets S and T. We define set

$$A = \{(i,j) | (i,j) \in C \wedge i \in S \wedge j \in T\}$$

and set

$$B = \{(i,j) | (i,j) \in C \wedge i \in T \wedge j \in S\}.$$

We decrease the t labels of the vertices in T by a small amount ϵ; then, the change of the objective per unit of ϵ is

$$P(t, C) = -\sum_{(i,j)\in A} \mathcal{C}^-_{ij}(d_{ij}) + \sum_{(i,j)\in B} \mathcal{C}^+_{ij}(d_{ij})$$

where $d_{ij} = t_j - t_i$. If this change is negative, we can always obtain a smaller objective value by decreasing the t labels of the vertices in T.

Algorithm ConvexCost-Buffering

```
1. ComputeTimingAndSlack(G);
2. d_ij ← u_ij  ∀(i,j) ∈ E
3. c_ij ← −C⁻_ij(d_ij)  ∀(i,j) ∈ E ∧ S_ij < 0
4. c_ij ← 0  ∀(i,j) ∈ E ∧ S_ij ≥ 0
5. while there exist positive cycles in G
6.       Augment maximal flows using s as
         the source and t as the sink;
7.       Select a min-cut M;
8.       ∀e ∈ M: insert a buffer into e;
9.       UpdateTimingSlack(G);
10.      c_ij ← −C⁻_ij(d_ij) − f_ij  ∀(i,j) ∈ E ∧ S_ij < 0
11.      c_ij ← 0  ∀(i,j) ∈ E ∧ S_ij ≥ 0
```

Fig. 2. Convex cost-flow-based buffering algorithm with the simplified separable model.

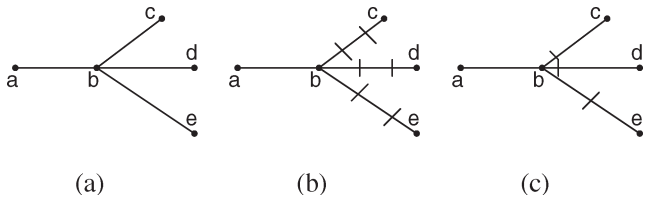

Fig. 3. Sensitivity computation problem. The short line segments represent buffers.

Thus, for the optimal solution, this change should be nonnegative for all the cuts. Define

$$\lambda(t) = \max\left(0, -\min_{\text{all cut } C} \frac{P(t, C)}{|C|}\right)$$

where $|C|$ is the number of edges in the cut C. Karzanov and Mccormick [6] have proved that t is optimal if and only if (iff) $\lambda(t) = 0$. They also proved the following theorem.

Theorem 1: A real λ is an upper bound on $\lambda(t)$ iff there exists a vector $x \in \mathcal{R}^E$ such that

$$\sum_{(i,j)\in E} x_{ij} - \sum_{(j,i)\in E} x_{ji} = 0 \qquad \forall i \in V; \quad \mathcal{C}^+_{ij}(d_{ij}) - x_{ij} \geq -\lambda$$

and

$$x_{ij} - \mathcal{C}^-_{ij}(d_{ij}) \geq -\lambda.$$

Thus, if $\lambda = 0$, we have

$$\mathcal{C}^+_{ij}(d_{ij}) \geq x_{ij} \quad \mathcal{C}^-_{ij}(d_{ij}) \leq x_{ij}.$$

Now, we obtain all these conditions.

Equation (1) shows that $t_j = t_i + d_{ij}$, but the delay of a gate or a wire may be discrete; thus, this equation is hard, if not impossible, to satisfy in reality. Thus, we relax it to $t_j \geq t_i + d_{ij}$. If this relaxed condition is satisfied, the timing constraint is satisfied.

Now, we design a heuristic to handle the buffering problem for the simplified model. The pseudocode of this algorithm, called ConvexCost-Buffering, is shown in Fig. 2. Let c_{ij}, f_{ij}, and S_{ij} denote the capacity, the flow, and the slack of the edge (i, j), respectively. Note that when the flow goes through an edge, we do not introduce a

TABLE I
COMPARISON RESULTS OF COSTBIN, CUTBIN, AND [9]

circuit			[9]		CutBIN				CostBIN		
Name	# Module	# Wire	# Buffer	Time (s)	# Buffer	Time (s)	Reduction	Speed up	# Buffer	Time (s)	Reduction
c1	22	98	172	0	146	0.01	15%	1×	127	0.01	26%
c2	44	197	305	5	176	0.02	43%	250×	145	0.01	52%
c3	81	398	606	16	306	0.06	50%	266.67×	276	0.04	54%
c4	159	799	887	10	464	0.14	48%	71.43×	449	0.11	49%
c5	258	1037	1096	28	767	0.25	30%	112.00×	709	0.34	35%
c6	505	2039	2140	20	1251	1.04	42%	19.23×	1137	1.80	47%
c7	2514	10039	10297	170	5612	20	45%	8.50×	5206	40	49%
c8	5034	20038	21201	344	10059	58	53%	5.93×	9403	142	56%
Average							41%				46%

residual backward edge, so the flows always go through the forward edges. Since x is the network flow, (3) is always satisfied. When a new buffer is inserted into a wire e, the delay of e decreases. Based on our simplified model, the delays of the fan-in edges of e do not change. Thus, when a buffer is inserted into each wire in the min-cut found by the algorithm, the maximal delay decreases. Eventually, the relaxed (1) is satisfied. Note that when we insert a buffer into a wire e, the existing k buffers on e also move such that the delay of e with $k+1$ buffers is minimized. This is the wire substitution idea introduced at the beginning of this section. The buffers are always inserted into the wires in the min-cut, so the *accumulated* flow x_{ij} on any wire (i,j) in the min-cut reaches the capacity $(-\mathcal{C}_{ij}^-(d_{ij}))$ and is not less than $(-\mathcal{C}_{ij}^+(d_{ij}))$. For the other wires, $-\mathcal{C}_{ij}^+(d_{ij}) \le x_{ij} \le -\mathcal{C}_{ij}^-(d_{ij})$ is obviously satisfied. Thus, (2) is also satisfied. Note that according to (1), condition (2) is equivalent to $-\mathcal{C}_{ij}^+(t_j - t_i) \le x_{ij} \le -\mathcal{C}_{ij}^-(t_j - t_i)$. However, since (1) is relaxed to an inequality in this part, these two conditions are not any more equivalent.

ConvexCost-Buffering computes the flows incrementally: the flow through an edge before one iteration is the sum of the flows through this edge in the previous iterations. Intuitively, if we do not consider the existing flows, i.e., the capacity of any edge (i,j) is set to $-\mathcal{C}_{ij}^-(d_{ij})$ instead of the residual capacity $-\mathcal{C}_{ij}^-(d_{ij}) - f_{ij}$, the number of inserted buffers might not be big, either. Based on this, we design a *greedy* variant of ConvexCost-Buffering based on min-cut, called MinCut-Buffering, for comparison.

B. Implementation Issues

In reality, the delay of a wire or a module is a function of its own parameters (e.g., output capacitance and intrinsic delay) and the *load*. Thus, the number of buffers needed on a wire also depends on the load; that is, $\mathcal{C}_{ij}(d_{ij})$ may have different values for different loads. Since the loads are not constant during the buffering process, $\mathcal{C}_{ij}$ is a function of the delays of both the wire (i,j) and its fan-outs. Thus, it is impossible to obtain a *separable* objective function. In this section, we consider how to deal with these issues.

The general framework of network-flow-based buffering algorithms is denoted as NetworkBIN. At the beginning, there is no buffer in the circuit. We use Ford–Fulkerson algorithm [5] to compute the maximal flow from s to t and, thus, find the min-cut. The significant difference between NetworkBIN and the previous two algorithms for the simplified model is that when a new buffer is inserted into a wire (line 8 in Fig. 2), NetworkBIN inserts buffers immediately after the other branches to decouple all the other branches.

We use the Elmore delay model for wires, modules, and buffers, as shown in [9]. However, our framework can be applied to other more general models. Based on the results in [9], the maximal delay change of wire (i,j) by inserting a new buffer into it, which is denoted as δ_{ij}, can be computed. The *delay sensitivity* of component (i,j) is defined as $-1/\delta_{ij}$. When the relations between component delays are

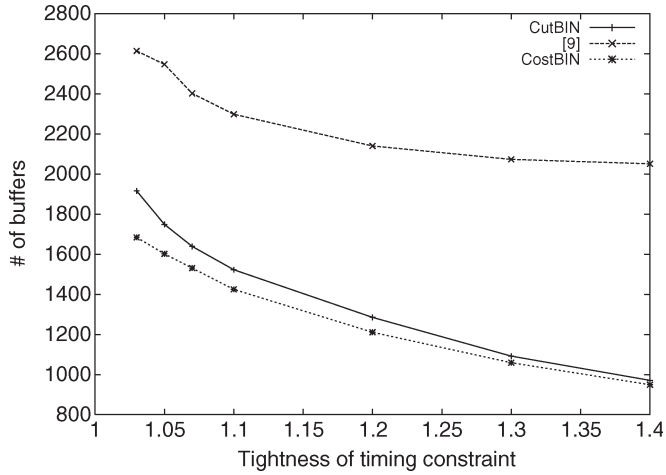

Fig. 4. Influence of the tightness of timing constraint on the number of inserted buffers.

considered, the sensitivity $(-\mathcal{C}_{ij}^-(d_{ij}))$ computation becomes difficult. As an example shown in Fig. 3(a), a net in a circuit has four pins. The slacks on the wires (b,c), (b,d), and (b,e) are much different, and these slacks are all negative. If NetworkBIN finds a min-cut through (b,c), (b,d), and (b,e), it will insert one buffer into each wire. After that, their slacks might still be negative, so more buffers need to be inserted. This solution is shown in Fig. 3(b). However, we may find a better solution by decreasing the delay of the wire (a,b) through decoupling some less critical branches: Since the slacks of the branches are different, we may only need to insert one buffer into the branch with the most negative slack and insert buffers into other branches to decouple them. This solution is shown in Fig. 3(c). The reason that NetworkBIN generates worse solutions is that the sensitivity computation is based on the assumption that no buffer will be inserted into all the other wires, but actually, the sensitivities of wires might change when new buffers are inserted into other branches. An approach to avoid this problem is to enforce that the min-cut contains, at most, one wire for each net, and the sensitivity computation also considers the delay changes of the fan-in edges introduced by the decoupling buffers. Thus, during the computation of the delay sensitivity, we always assume that all the other branches are decoupled. In addition, in each iteration, for each net, we set the capacity of the branch with the most negative slack (if there are more than one branch having the most negative slack, pick either one of them) to be its delay sensitivity and set the capacities of other branches to be 0. In the example of Fig. 3(a), when we compute the delay sensitivity of (b,e), we assume that there are buffers immediately after the branches (b,c) and (b,d). Based on this assumption, we compute the delay changes of the wires (a,b) and (b,e) when a new buffer is inserted into (b,e). If the branch (b,e) with the most negative slack is

TABLE II
COMPARISON RESULTS OF COSTBIN, CUTBIN, NETBIN, AND [8]

circuit			[8]		CostBINv1				CostBINv2	NetBIN	CutBIN	
Name	# Cand	#Segs $S < 0$	Area	Time (s)	Area	Time (s)	Reduction	Speed up	Area	Area	Area	Time (s)
c1	695	689	598.5	151	333.5	0.17	44%	888×	301.5	629.0	340.5	0.17
c2	1278	1269	659.0	90	368.0	0.37	44%	243×	321.5	773.5	381.0	0.40
c3	2564	2564	1955.0	256	643.5	2.58	67%	99×	560.5	1473.0	660.0	2.98
c4	5168	4821	2636.0	979	1089.0	5.54	59%	177×	940.0	3096.0	1092.0	8.49
c5	5579	5507	2933.5	859	1594.5	16.43	46%	52×	1414.0	-	1668.0	11.05
c6	11163	11126	4842.5	1855	2604.0	32.83	46%	57×	2262.5	-	2762.0	25.00
c7	53612	53561	24272.0	4662	11234.0	682.73	54%	7×	-	-	11823.0	555.51
c8	107931	107847	50004.0	18550	21191.5	2399.47	58%	8×	-	-	22418.0	1716.93
Average							52%					

in the found min-cut, we insert a buffer into (b, e), and decouple the branches (b, c) and (b, d) by inserting buffers near the split point of these branches, thus obtaining a solution, as shown in Fig. 3(c).

We design two variants of the NetworkBIN framework: one considers the existing flows, whereas the other does not.

We first present our convex cost-flow-based buffering algorithm, called CostBIN, which is a practical version of the ConvexCost-Buffering. The slacks and timing information are computed in the step ComputeTimingAndSlack, with the timing constraint as the required time at sink t. The capacity setting step in CostBIN works as follows: For any edge (i, j), if $S_{ij} \geq 0$, it should be excluded from the network; thus, its capacity is 0. To avoid the previously discussed sensitivity computation problem, CostBIN assumes that all the other wire branches are decoupled in the delay sensitivity computation step. Let output$((i, j))$ represent the set containing all the fan-out edges of (i, j); then, the maximal change of the sum of the delays of (i, j) and its fan-in edge (k, i) is

$$T_{ij} = \delta_{ij} + \hat{R}_{ij}\left(\#\left(\text{output}\left((k,i)\right)\right) C_b + Cp_{ij} - \hat{C}_{ki}\right)$$

where function $\#(S)$ represents the number of elements in set S, $\hat{R}_{ij}$ represents the upstream resistance at the input of wire (i, j), $\hat{C}_{ki}$ represents the downstream capacitance at the output of wire (k, i), p_{ij} represents the wire length from the input of wire (i, j) to the first buffer of the wire, C_b represents the input capacitance of a buffer, and C represents the capacitance of a unit-length wire. If $T_{ij} < 0$, the capacity is $-1/T_{ij} - f_{ij}$; otherwise, it is infinity. Since δ_{ij} is 0 for any edge in the forbidden area, its capacity is infinity.

If we do not consider the existing flows on edges, we have a greedy version of CostBIN. This variant is called CutBIN, which is a practical version of the MinCut-Buffering. An approach to speed up the algorithm is to restrict the flow in a subnetwork that contains only the most critical paths. CutBIN uses the current maximal delay from s to t as the required time at sink t and computes the slacks of components. A subgraph containing only the components with slacks less than S_{th} is called the *critical subgraph* of the circuit. In addition to the speeding-up effect, S_{th} can also avoid the sensitivity computation problem. S_{th} is a user-specified value or is adaptively computed using some heuristics [3]. Note that this critical subgraph technique can also be used in CostBIN.

C. Extensions

Now, we extend these algorithms to handle more practical issues: the buffers have several different sizes, and the candidate locations for buffering are specified. The objective is to minimize the total area of buffers instead of the buffer number.

The buffering candidate locations and the Steiner points split the wires into many segments. Thus, the buffering locations can only be at the starting point of segments. For each segment (i, j) with a buffering candidate location, we can easily compute the delay difference T_{ij} when a new buffer is inserted or the original buffer is replaced by a bigger buffer at the starting point. The capacity in CostBIN is set to be $|\Delta_S/T_{ij}| - f_{ij}$, where Δ_S is the buffer size difference. We can similarly set the capacities in CutBIN. The buffers with the minimal size are always selected for the decoupling buffers.

When the timing constraint is satisfied, we can refine the solution to reduce more cost. Because the required arrival time at each sink of each net is currently known, for each net, particularly the large nets, Lillis's min-cost buffering [7] can be used to find a solution that has a minimal buffer area and satisfies the timing requirement. It is obvious that this solution has, at most, the cost of the solution before the refinement. However, note that even with speed-up techniques, as shown in [12], the min-cost buffering still introduces much memory overhead if the circuit has big nets.

III. EXPERIMENTAL RESULTS

We have implemented the CostBIN algorithms in C. We use the parameters from the 100-nm technology [4]. We obtained four test cases from Liu *et al.* and generated additional four cases using the case generator in [9]. All experiments are run on a Linux PC with a 2.4-GHz Xeon central processing unit and 2.0-GB memory.

To test the network-flow-based buffer insertion, we obtained the source code of the algorithms in [8] and [9] from Liu *et al.* for comparison and also implemented the CutBIN algorithm for comparison. The comparison results of CostBIN, CutBIN, and [9] are shown in Table I. We set the timing constraints to be 0.2 times greater than the minimal delays from s to t that can be achieved by buffering, which are computed by the min-delay buffering algorithm in [9]. The results show that CutBIN achieves a 41% reduction on the number of buffers on average as compared with [9] and is much more efficient than [9], and CostBIN achieves a 46% reduction on the number of inserted buffers on average as compared with [9]. We can also see that CostBIN always inserts fewer buffers than CutBIN.

Using the case "c6" with 505 modules and 2039 wires, we test the influence of the tightness of timing constraint on the number of inserted buffers. As shown in Fig. 4, both CostBIN and CutBIN save even more buffers than [9] when the timing constraints are looser, indicating that they are more effective buffering approaches. In particular, CostBIN always inserts fewer buffers than CutBIN under the same timing constraint, and when the timing constraint is tighter, CostBIN saves more buffers than CutBIN does. Note that our approaches may not obtain a valid solution when the tightness is 1.0. Since we have not introduced the backward flows in our approaches, if our approaches make the wrong decision in one iteration, they do not have the opportunity to correct it, so the valid solution cannot be found. This is a limitation we plan to solve in our future work. Because of the interaction of the delays of components, how to obtain a converged solution when the backward flows are introduced is still an open problem.

The dynamic programming-based work [8] extends the Lagrangian relaxation-based buffering algorithm in [9] to handle the cases where

there are multiple types of buffers and the candidate locations for buffering are specified. We also implemented a net-based buffering approach, called NetBIN. NetBIN buffers nets one by one until the timing constraint is satisfied. We also use a refinement step at the end of NetBIN to reduce the buffers, and our experience is that the refinement can reduce the buffers by at least 50%. Table II shows the comparison results of CostBIN, CutBIN, NetBIN, and [8]. There are four types of buffers, and the buffering candidate locations are specified. The timing constraints are 0.2 times greater than the minimal achievable delays from s to t by buffering, computed by the min-delay buffering algorithm in [8]. We observed that NetBIN inserts more buffers than [8] for most of those cases. NetBIN does not budget the timing in a global view, so the required time at sinks may mislead the buffering. Thus, buffering each net such that the required time at the root of the net is maximized does not guarantee that the delay from s to t is minimized. For example, for the last four test cases, it is difficult for NetBIN to obtain a feasible solution in 5 h. We also observed that the final solutions computed by the min-cost buffering algorithm in [8] often do not satisfy the timing constraint, so we select the solution that satisfies the timing constraint and has the minimal buffer area *during* the iterations. We also implemented the refinement step in CostBIN. The CostBIN without the refinement is denoted as CostBINv1, and the CostBIN with the refinement is denoted as CostBINv2. Column 3 shows the number of wire segments with negative slacks. Note that the wires in these test cases are separated into many small wire segments in this part, and the number of buffering locations (shown in column 2) is equal to the number of wire segments. The results indicate that CostBINv1 achieves a 52% reduction on the total area of buffers on average. The refinement step in CostBINv2 reduces an additional 12% of the buffering area on average, but "c7" and "c8" cannot be finished for CostBINv2 because of the big memory overhead. Also, the CostBIN inserts fewer buffers than the CutBIN in this situation.

REFERENCES

[1] C. J. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion," in *Proc. Des. Autom. Conf.*, 1997, pp. 588–593.

[2] C. J. Alpert, M. Hrkic, and S. T. Quay, "A fast algorithm for identifying good buffer insertion candidate locations," in *Proc. Int. Symp. Phys. Des.*, 2004, pp. 47–52.

[3] R. Chen and H. Zhou, "Efficient algorithms for buffer insertion in general circuits based on network flow," in *Proc. Int. Conf. Comput.-Aided Des.*, 2005, pp. 322–326.

[4] J. Cong and Z. Pan, "Interconnect performance estimation models for design planning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 6, pp. 739–752, Jun. 2001.

[5] J. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton, NJ: Princeton Univ. Press, 1962.

[6] A. V. Karzanov and S. T. McCormick, "Polynomial methods for separable convex optimization in unimodular linear spaces with applications," *SIAM J. Comput.*, vol. 26, no. 4, pp. 1245–1275, 1997.

[7] 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*, vol. 31, no. 3, pp. 437–447, Mar. 1996.

[8] I.-M. Liu, A. Aziz, and D. F. Wong, "Meeting delay constraints in DSM by minimal repeater insertion," in *Proc. DATE*, 2000, pp. 436–440.

[9] I.-M. Liu, A. Aziz, D. F. Wong, and H. Zhou, "An efficient buffer insertion algorithm for large networks based on Lagrangian relaxation," in *Proc. Int. Conf. Comput. Des.*, 1999, pp. 210–215.

[10] P. Saxena, N. Menezes, P. Cocchini, and D. A. Kirkpatrick, "The scaling challenge: Can correct-by-construction design help?" in *Proc. Int. Symp. Phys. Des.*, 2003, pp. 51–58.

[11] W. Shi and Z. Li, "A fast algorithm for optimal buffer insertion," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 6, pp. 879–891, Jun. 2005.

[12] W. Shi, Z. Li, and C. J. Alpert, "Complexity analysis and speedup techniques for optimal buffer insertion with minimum cost," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2004, pp. 609–614.

[13] C. N. Sze, C. J. Alpert, J. Hu, and W. Shi, "Path based buffer insertion," in *Proc. Des. Autom. Conf.*, 2005, pp. 509–514.

[14] L. P. P. P. van Ginneken, "Buffer placement in distributed *RC*-tree networks for minimal Elmore delay," in *Proc. Int. Symp. Circuits Syst.*, 1990, pp. 865–868.

[15] H. Zhou, D. F. Wong, I.-M. Liu, and A. Aziz, "Simultaneous routing and buffer insertion with restrictions on buffer locations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 7, pp. 819–824, Jul. 2000.

λ-OAT: λ-Geometry Obstacle-Avoiding Tree Construction With $O(n \log n)$ Complexity

Tom Tong Jing, Zhe Feng, Yu Hu, Xianlong L. Hong, Xiaodong D. Hu, and Guiying Y. Yan

***Abstract*—Obstacle-avoiding rectilinear Steiner minimal tree (OARSMT) construction is an essential part of routing. Recently, IC routing and related researches have been extended from Manhattan architecture (λ_2-geometry) to Y-/X-architecture (λ_3-/λ_4-geometry) to improve the chip performance. This paper presents an $O(n \log n)$ heuristic, λ-OAT, for obstacle-avoiding Steiner minimal tree construction in the λ-geometry plane (λ-OASMT). In this paper, based on obstacle-avoiding constrained Delaunay triangulation, a full connected tree is constructed and then embedded into λ-OASMT by zonal combination. To the best of our knowledge, this is the first work addressing the λ-OASMT problem. Compared with most recent works on OARSMT problem, λ-OAT obtains up to 30-Kx speedup with quality solution. We have tested randomly generated cases with up to 10 K terminals and 10-K rectilinear obstacles within 4 seconds on a Sun V880 workstation (755-MHz CPU and 4-GB memory). The high efficiency and accuracy of λ-OAT make it extremely practical and useful in the routing phase.**

***Index Terms*—Physical design, routing, Steiner tree, very large scale integration (VLSI).**

I. INTRODUCTION

Routing a net, finding a rectilinear Steiner minimal tree (RSMT) for a given terminal set, is a fundamental problem in physical design. In practical routing applications, macrocells, IP blocks, and prerouted nets are often regarded as obstacles. Thus, obstacle-avoiding rectilinear Steiner minimal tree (OARSMT) construction is often used as an accurate wire length even the delay estimation throughout the process of routing. It was proved that the RSMT problem is NP-complete [1].

Manuscript received April 30, 2006; revised December 18, 2006. This work was supported by the Key Project of Chinese Ministry of Education under Grant 106008, by the Specialized Research Fund for the Doctoral Program of Higher Education (SRFDP) of China under Grant 20050003099, and by the NSFC under Grant 90607001. This paper was recommended by Associate Editor P. H. Madden.

T. T. Jing was with the Computer Science and Technology Department, Tsinghua University, Beijing 100084, China. He is now with the Electrical Engineering Department, University of California at Los Angeles (UCLA), Los Angeles, CA 90095 USA (e-mail: tomjing@ucla.edu).

Z. Feng and X. L. Hong are with the Computer Science and Technology Department, Tsinghua University, Beijing 100084, China.

Y. Hu is with the Electrical Engineering Department, University of California at Los Angeles (UCLA), Los Angeles, CA 90095 USA.

X. D. Hu and G. Y. Yan are with the Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing 100080, China.

Digital Object Identifier 10.1109/TCAD.2007.896291