Provably Good Global Buffering Using an Available Buffer Block Plan*

Feodor F. Dragan, Andrew B. Kahng, Ion Măndoiu†, Sudhakar Muddu‡, and Alexander Zelikovsky¶

UCLA Department of Computer Science, Los Angeles, CA 90095-1596
†College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280
‡Silicon Graphics, Inc., Mountain View, CA 94039
¶Department of Computer Science, Georgia State University, Atlanta, GA 30303
{dragan,abk}@cs.ucla.edu, mandoiu@cc.gatech.edu, muddu@mti.sgi.com, alexz@cs.gsu.edu

Abstract

To implement high-performance global interconnect without impacting the performance of existing blocks, the use of buffer blocks is increasingly popular in structured-custom and block-based ASIC/SOC methodologies. Recent works by Cong et al. [6] and Tang and Wong [25] give algorithms to solve the *buffer block planning* problem. In this paper we address the problem of how to perform buffering of global nets *given an existing buffer block plan*. Assuming as in [6, 25] that global nets have been already decomposed into two-pin connections, we give a provably good algorithm based on a recent approach of Garg and Könemann [8] and Fleischer [7]. Our method routes connections using available buffer blocks, such that required upper and lower bounds on buffer intervals – as well as wirelength upper bounds per connection – are satisfied. Unlike [6, 25], our model allows more than one buffer to be inserted into any given connection. In addition, our algorithm observes buffer parity constraints, i.e., it will choose to use an inverter or a buffer (= co-located pair of inverters) according to source and destination signal parity. The algorithm outperforms previous approaches [6] and has been validated on top-level layouts extracted from a recent high-end microprocessor design.

1 Introduction

A key consequence of the semiconductor technology roadmap [23] is the dominant effect of interconnect in deep-submicron design. As clock frequencies reach and exceed the gigahertz level, each top-level global net must undergo repeater insertion (among other optimizations; see [5, 18, 20]) to maintain signal integrity and reasonable signal delay.[1] Estimates of the need for repeater insertion range up to $O(10^6)$ repeaters for top-level on-chip interconnect when we reach the 50nm technology node. These repeaters are large (anywhere from $40\times$ to $200\times$ minimum inverter size), affect global routing congestion, can entail non-standard cell height and special power routing requirements, and can act as noise sources. In a block- or reuse-based methodology, designers seek to isolate repeater for global interconnect from individual block implementations.

For these reasons, a *buffer block* methodology has become increasingly popular in structured-custom and block-based ASIC/SOC methodologies. Two recent works by Cong et al. [6] and Tang and Wong [25] give algorithms to solve the *buffer block planning* problem. Their buffer block planning formulation is roughly stated as: Given a placement of circuit blocks, and a set of two-pin connections with a *feasible region* for a *single* buffer per connection, plan the shape and location of *buffer blocks* so as to maximally use available free space and minimally impact the existing floorplan.

The formulation of [6, 25] requires a single buffer per connection. This allows use of computational geometry and network flows with respect to *feasible regions*, and the buffer block plan implicitly contains the global buffering solution for the netlist of connections. However, in reality, multiple buffers are often needed per connection. For example, global repeater rules for a high-end microprocessor design in 0.25μm CMOS [14] require repeater intervals of at most 4500μm.[2] The number of buffers needed for a given connection depends strongly on the length of the connection; as noted in [14], the repeater interval is not only required for delay reduction, but also for crosstalk noise immunity and edge slewtime control. We also note that buffer block resources may not always be completely plannable – buffer sites are often embedded in IP blocks or in block "collars" within a hierarchical ASIC/SOC methodology.

In this paper, we address the problem of how to perform buffering of global nets *given an existing buffer block plan*. (Hence, our work is compatible with and complements the methods in [6, 25].) Assuming as in [6, 25] that global nets have been already decomposed into two-pin connections, we give a provably good algorithm based on a recent approach of Garg and Könemann [8] and Fleischer [7]. Our method routes connections using available buffer blocks, such that required upper and lower bounds on repeater interval – as well as length upper bounds per connection – are satisfied. Notably, unlike [6, 25] our model allows *more than one buffer* to be inserted into any given connection. In addition, our algorithm observes *repeater parity constraints*, i.e., it will choose to use an inverter or a buffer (= co-located pair of inverters) according to source and destination signal parity. Informally, our problem is defined as follows.

Given:

- a planar region with rectangular obstacles;
- a set of source-destination pairs (point pairs) in the region;
- each pair has a parity requirement;
- each (timing-driven) source-destination pair has a limit on the length of its path, i.e., there is a prescribed bound that limits the number of used repeaters;
- a set of buffer blocks, each with given capacity; and
- an interval $[L,U]$ that defines lower and upper bounds on the distance between repeaters.

Global Buffering Problem: Construct a route for each source-destination pair, such that each route passes through zero or more repeaters, subject to:

- the distance between the source of a route and its first repeater is between L and U;
- the distance between any two consecutive repeaters on any given route is between L and U;[3]
- the distance between the last repeater on a route and the route's destination is again between L and U;
- the number of routes passing through any given buffer block does not exceed that block's capacity;

*This work was partially supported by Cadence Design Systems, Inc. and the MARCO Gigascale Silicon Research Center. Andrew B. Kahng is now Professor of Computer Science and Engineering, and of Electrical and Computer Engineering, at the University of California, San Diego. Feodor Dragan is now Associate Professor of Mathematics and Computer Science at Kent State University. Sudhakar Muddu is now with Sanera Systems, Inc.

[1] Following the literature, we will use the terms *buffer* and *repeater* fairly interchangeably. When we need to be more precise: a repeater can be implemented as either an inverter or as a buffer (= two co-located inverters).

[2] Chip side length is now routinely at 20000μm. Multiple repeaters per connection are also implicit in the cycle time modeling of the technology roadmap [23].

[3] Our approach can also handle the case where L and U bounds differ for different source-destination pairs, and depend on the position (from source) of two consecutive repeaters on the routing path.

- the number of repeaters on each source-destination route should be of the given parity (if necessary, to achieve the parity condition, the route can use two sites in a single buffer block, i.e., a buffer instead of an inverter; in this case we use *two* units of the block's capacity);
- the number of repeater sites on each source-destination route should not exceed the given limit (upper bound).[4]

Note that coupling awareness is built into the user-prescribed repeater interval, via the switch-factor methodology [13] which accounts for best- and worst-case Miller coupling between adjacent nets. Typically, switch factors between 0 and 2, or between -1 and 3, are used to multiply the nominal victim-aggressor coupling capacitance when doing timing analysis. Then, repeater intervals are calculated to maximize interconnect performance, subject to bounds on noise and delay uncertainty [12].

The most simple-minded implementation of our approach will treat all source-sink paths as "equally critical": buffers are inserted at the same regular interval along each source-sink path, independent of the path's length. However, we note that if a particular source-sink path is not timing-critical, our method allows net ordering that would process it last, or net weighting that would decrease the priority of its claim on buffer resources. Post-processing of the solution could remove inserted buffers (starting at most-used blocks) until further removal violates some actual timing budget. On the other hand, in early chip planning stages it is typical for aggressive or "optimal" buffering to be performed, so that all global paths are as fast as possible [9]. This breaks the chicken-egg problem of budgeting between-block and within-block paths in pre-synthesis RTL planning; it also allows maximum timing budgets for within-block timing paths.[5]

Our paper is organized as follows. In Section 2, we reduce the Global Buffering Problem to a generalized version of integer multicommodity flow (MCF), and in Section 3 we give algorithms for approximating the optimal fractional relaxation of this MCF – for both *maximum-routing* and *minimum-routing-cost* formulations – within any desired accuracy. Our algorithm is based on recent results of [8] and [7], which we extend to a vertex-capacitated context. Section 4 describes a randomized rounding procedure that converts near-optimal fractional MCF solutions to near-optimal integral solutions. MCF based heuristics (with or without randomized rounding) have been applied to VLSI global routing [24, 3, 11]. However, global routing appears less naturally suited for MCF than our Global buffering problem; it does not seem to yield as strong theoretical bounds nor as effective implementations. Section 5 describes the four Global Buffering heuristics that we have implemented: (i) Greedy, (ii) ε-approximate MCF (ε-MCF), (iii) Greedy enhancement of ε-MCF (ε-MCFG), and (iv) "1-Shot". Finally, Section 6 gives the results of these heuristics on test cases extracted from top-level layout of a recent high-end microprocessor, and Section 7 concludes with a list of open research directions.

2 Integer Multicommodity Flow Formulation

Assume that we have K pairs of *terminals* (s_k,t_k), and n *buffer blocks* $\{r_1,\ldots,r_n\}$. Denote $S=\{s_1,\ldots,s_K\}$, $T=\{t_1,\ldots,t_K\}$, $R=\{r_1,\ldots,r_n\}$. Let also $c(r)\in\mathbf{N}$ denote the *capacity of the buffer block* $r\in R$, $a_k\in\{$*even, odd*$\}$ be the *parity requirement* for pair (s_k,t_k), and l_k be the prescribed upper bound on the number of buffers on path between source s_k and destination t_k.

We construct a graph $G=(V,E)$ as follows. Let p_{xy} be a rectilinear path connecting points x and y of a planar region that avoids all rectangular obstacles given in the region. Denote by $d(x,y)$ the length of a shortest such path. The vertex set V of G is $S\cup T\cup R$. The edge set E contains all edges of type vv, $v\in R$, (such an edge is called a loop), as well as all edges xy for which $L\le d(x,y)\le U$. (This graph can be constructed using, e.g., shortest paths in visibility graphs [22].)

We will use the following definition of an (s_k,t_k)-path. A path $p=(s_k,v_1,v_2,\ldots,v_l,t_k)$ in G between the k^{th} source s_k and sink t_k is an (s_k,t_k)-*path* if

- $v_i\in R$ for each $i=1,\ldots,l$,
- there exists at most one pair of different indices $i,j\in\{1,\ldots,l\}$ such that $v_i=v_j$ (i.e., v_i and v_j represent the same buffer block), and in this case we must have $|i-j|=1$,
- the parity of l is a_k,
- $l\le l_k$.

Having this graph and the definition of (s_k,t_k)-path, our problem is as follows. For every pair of terminals (s_k,t_k), $k=1,\ldots,K$, find an (s_k,t_k)-path, subject to the constraint that the number of times any vertex $r\in R$ is used in these paths should not exceed the capacity $c(r)$ of r. If the problem has no solution, we want to maximize the number of pairs (s_k,t_k) that can be connected without violating capacity, parity and timing-driven constraints. Let us call this problem the *Maximum Routing-Via-Buffer-Blocks* (**MRVBB**) *problem.*

The **MRVBB** problem can be formulated as a maximum integral multicommodity flow problem on a graph with vertex capacities. An instance consists of a graph $G=(V,E)$ with vertex capacities $c:V\to\mathbf{N}$ and K pairs of terminals (s_k,t_k), with one commodity associated with each pair. We seek a multicommodity flow such that the sum of the flows of all commodities is maximized.

Define capacities on all vertices of G as follows:

$$c(v):=\begin{cases}1 & \text{if}\quad v\in S\cup T,\\ \textit{capacity of buffer block} & \text{if}\quad v\in R.\end{cases}$$

Let P_k be the set of all possible (s_k,t_k)-paths. We define P to be the union of all P_k, i.e., $P=\bigcup_{k=1}^{K}P_k$. For $p\in P$, let f_p be a variable denoting the flow along this path. The commodity to which this flow corresponds is clear from looking at indices of the end vertices of p. Thus, we have the following integer linear program:

$$\begin{array}{ll}\textbf{maximize} & \sum_{p\in P} f_p\\ \textbf{subject to} & \sum_{p\in P} q_p(v)f_p\le c(v) \quad \forall v\in V\\ & f_p\in\{0,1\} \quad \forall p\in P.\end{array}$$

where

$$q_p(v):=\begin{cases}0 & \text{if}\quad v\notin p,\\ 1 & \text{if}\quad v\in p,\text{but } vv \text{ is not a loop on } p,\\ 2 & \text{if}\quad v\in p, \text{ and } vv \text{ is a loop on } p.\end{cases}$$

Rather than solve the integer program directly, we consider the LP relaxation, substituting the last constraint by

$$f_p\ge 0 \qquad \forall p\in P.$$

We call the problem of finding an optimal solution to this LP the *Maximum Fractional Routing-Via-Buffer-Blocks* (**MFRVBB**) *problem.* After solving the fractional routing problem, we will apply randomized rounding to get an approximate solution for the initial **MRVBB** problem. (Note that, in fact, this MCF formulation can also handle routing congestion, e.g., by putting on each edge of the graph G a new vertex (pseudo buffer block) with suitable capacity.) The dual of this LP is

$$\begin{array}{ll}\textbf{minimize} & \sum_{v\in V} w(v)c(v)\\ \textbf{subject to} & \sum_{v\in p} w(v)\ge 1 \quad \forall p\in P\\ & w(v)\ge 0 \quad \forall v\in V.\end{array}$$

The dual can be viewed as an assignment of non-negative weights, $w(\cdot)$, to the vertices of G such that the weight of any

[4] Our approach also handles the variant where each source-destination pair has a weight (importance or criticality of each pair) and a measure of cost is to be minimized.

[5] Thus, blocks can go through synthesis, place and route with more aggressive area targets. A strategy of uniform buffering of as many global nets as possible also helps control signal integrity and delay uncertainty issues.

path $p \in P$ is at least 1; the objective is to minimize the sum $\sum_{v \in V} w(v)c(v)$. Here, the *weight of the path* is the sum of the weights of vertices forming this path (since p is by definition a sequence of vertices, not a set of vertices, if the path uses a loop vv then vertex v contributes twice to the weight of the path). The approximation algorithm given in the next section simultaneously solves both primal and dual problems – the dual solution is then used in proving the approximation guarantee of the algorithm.

3 Approximation of Node-Capacitated Fractional MCF

The fractional routing problem, **MFRVBB**, can be solved exactly in polynomial time by any polynomial-time LP algorithm. However, such algorithms are very inefficient in practice. Two recent results in approximation algorithms allow us to obtain high-quality solutions efficiently. Our algorithm for solving the **MFRVBB** problem is based on the fast approximate algorithm for the maximum multicommodity flow problem on graphs with edge capacities due to Garg and Könemann [8]. Small modifications allow us to adapt their method to our context of vertex-capacitated graphs, source-sink path parity constraints and timing-driven constraints. Moreover, we apply an idea of Fleischer [7] to reduce the number of minimum weight path computations made by the algorithm.

Denote $D(w) = \sum_{v \in V} w(v)c(v)$ and let $\alpha(w)$ be the weight of a minimum weight path from P (with respect to $w(\cdot)$). The dual problem is equivalent to finding a weight function $w : V \to \mathbf{R}^+$ such that $\beta = \frac{D(w)}{\alpha(w)}$ is minimized (see [8]). Algorithm 1 solves the **MFRVBB** problem for any given approximation ratio.

Algorithm 1: MFRVBB Algorithm

Input: Graph G with source-sink pairs (s_k, t_k), node capacities $c(v)$
Output: Flows $f_k(v) \in [0,1]$, $k = 1, \ldots, K$, $v \in V(G)$ satisfying capacity constraints

1. Set $f = 0$.
2. Set $w(v) = \delta$ for all $v \in V$.
3. Set $f_k(v) = 0$ for all $v \in V$ and $k = 1, \ldots, K$.
4. For $i = 1$ to $\log_{1+2\varepsilon} \frac{1+2\varepsilon}{\delta}$ do
   For $k = 1$ to K do
   Find a path p in P_k with minimum weight w.r.t. $w(\cdot)$.
   While $weight(p) < min\{1, \delta(1+2\varepsilon)^i\}$ do
   $f = f + 1$;
   For all $v \in p$, if p uses a loop vv then set $f_k(v) = f_k(v) + 2$ and $w(v) = w(v)(1 + \frac{2\varepsilon}{c(v)})$; else set $f_k(v) = f_k(v) + 1$ and $w(v) = w(v)(1 + \frac{\varepsilon}{c(v)})$.
   Find a path p in P_k with minimum weight w.r.t. $w(\cdot)$.
   End while
   End for
   End for
5. Output $\frac{f}{2\log_{1+2\varepsilon} \frac{1+2\varepsilon}{\delta}}$, and $\frac{f_k(v)}{2\log_{1+2\varepsilon} \frac{1+2\varepsilon}{\delta}}$ for each $v \in V$ and $k = 1, \ldots, K$.

In the algorithm, $f_k(v)$ denotes the flow of commodity k passing through a vertex v, and f denotes the total flow routed. The algorithm associates a weight with each vertex, and every time it routes an unit flow along some (s_k, t_k)-path p from P_k ($k = 1, \ldots, K$) it multiplies the weight of every vertex on this path by $1 + \varepsilon/c(v)$ for a fixed ε (if the path uses a loop vv, then the weight of v is multiplied by $1 + 2\varepsilon/c(v)$). Initially, every vertex v has weight δ for some constant δ. Thus, the heavier the vertex the greater the flow through it.

According to Garg and Könemann's approximate algorithm [8], we must route a unit flow along a lightest (with respect to current weight function $w(\cdot)$) path from P, if the weight of this path is less than 1. We also must stop after t iterations where t is the smallest number such that $\alpha(w)$, computed with respect to vertex weights $w(\cdot)$ of this iteration, is at least 1. Fleischer [7] noted that instead of finding the lightest path in P, one can settle for some path within a factor of $(1 + 2\varepsilon)$ of the lightest while obtaining a similar approximation guarantee.

Let $w_{i-1}(\cdot)$ be the weight function at the beginning of the i^{th} iteration. We have $w_0(v) = \delta$ for each $v \in V$. For brevity denote $\alpha(w_i)$, $D(w_i)$ by $\alpha(i)$, $D(i)$ respectively. Following Fleischer, we cycle through the commodities, sticking with a commodity until the lightest source-sink path for that commodity is above an $1 + 2\varepsilon$ factor times a lower bound estimate of the overall lightest path. Let $\bar{\alpha}(i)$ be a lower bound on $\alpha(i)$. To start, we set $\bar{\alpha}(0) = \delta$. As long as there is some $p \in P$ with $weight(p) \leq min\{1, (1+2\varepsilon)\bar{\alpha}(i)\}$, we augment flow along p. When this no longer holds, we know that the weight of the lightest path is at least $(1+2\varepsilon)\bar{\alpha}(i)$, and so we set $\bar{\alpha}(i+1) = (1+2\varepsilon)\bar{\alpha}(i)$. Thus, throughout the course of the algorithm, $\bar{\alpha}$ takes on values in the set $\{\delta(1+2\varepsilon)^i\}_{i \in N}$. Since $\alpha(0) \geq \delta$ and $\alpha(t-1) < 1$, $\alpha(t) < 1 + 2\varepsilon$. Thus, when we stop, $\bar{\alpha}(t)$ is between 1 and $1 + 2\varepsilon$. Each increase of $\bar{\alpha}$ is by an $1 + 2\varepsilon$ factor, hence the number of increases of $\bar{\alpha}$ is $\log_{1+2\varepsilon} \frac{1+2\varepsilon}{\delta}$ (and the final value of i is $\lfloor \log_{1+2\varepsilon} \frac{1+2\varepsilon}{\delta} \rfloor$).

Between updates to $\bar{\alpha}$, the algorithm proceeds by considering each commodity one by one. As long as the lightest path for commodity k has weight less than the minimum of $1 + 2\varepsilon$ times the current value of $\bar{\alpha}$ and 1, flow is augmented along such a lightest path. When $min_{p \in P_k} weight(p) \geq (1+2\varepsilon)\bar{\alpha}$, commodity $k+1$ is considered. After all K commodities are considered, $\bar{\alpha}$ is updated. A total of at most $K \log_{1+2\varepsilon} \frac{1+2\varepsilon}{\delta}$ minimum weight path computations are used to update α over the course of the algorithm.

Note also that the number of possible augmentations is at most $K \log_{1+2\varepsilon} \frac{1+2\varepsilon}{\delta}$. Indeed, at the start $w(v) = \delta$ for each vertex v. The last time the weight of a vertex is updated, it is on a path of weight less than one, and it is increased by at most a factor of $1 + 2\varepsilon$. Hence, the final weight of any vertex is at most $1 + 2\varepsilon$. Since every augmentation increases the weight of some sink t_k ($k = 1, \ldots, K$) by a factor of at least $1 + 2\varepsilon$, the number of possible augmentations is at most $K \log_{1+2\varepsilon} \frac{1+2\varepsilon}{\delta}$. This, together with our observation on the number of times $\bar{\alpha}$ is recomputed, implies a runtime of $O(Km \log_{1+2\varepsilon} \frac{1+2\varepsilon}{\delta})$ for our algorithm, where m is the number of edges between buffer blocks in our graph G.

Theorem 1 *Algorithm 1 yields a $(1 + 8\varepsilon)$-approximation for the* **MFRVBB** *problem by choosing $\delta = (1+2\varepsilon)((1+2\varepsilon)L)^{-\frac{1}{2\varepsilon}}$ and $\varepsilon < .07$, where L is the number of vertices in the longest simple path of G between any source-sink pair.*

Proof. Our proof is an extension of the proof of Fleischer [7] (see also [8]) to vertex-capacitated case, and is omitted due to space constraints. □

In Algorithm 1 we need to solve the following problem. Let G_k ($k = 1, \ldots, K$) be a subgraph of the graph G induced by vertices $\{s_k, t_k\} \cup R$ (recall that each vertex $v \in R$ has a loop $vv \in E$). Let also each vertex v of G_k have a non-negative weight $w(v)$. Find a minimum weight path p_k in G_k connecting s_k with t_k which passes through an even (odd, depending on a_k) number of vertices, with the number of vertices not exceeding l_k. This path may contain at most one loop. So, the vertex weight will contribute either once or twice (in case of loop) to the weight of the path.

We will reduce this problem to the usual shortest path problem on a edge-weighted directed acyclic graph (dag) D_k with $2 + nl_k$ vertices and at most $n + n^2(l_k - 1) + n\lceil l_i/2 \rceil$ arcs, constructed as follows:

- $V(D_k) = s_k \cup \{r_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq l_k\} \cup \{t_k\}$

- $E(D_k) = E_1 \cup E_2 \cup E_3$, where

$$\begin{aligned} E_1 &= \{(s_k, r_{i,1} \mid 1 \le i \le n, (s_k, r_i) \in E(G)\} \\ E_2 &= \{(r_{i,j}, r_{i',j+1}) \mid 1 \le i,i' \le n, 1 \le j < l_k, \\ &\quad (r_i, r_{i'}) \in E(G)\} \\ E_3 &= \{(r_{i,j}, t_k) \mid 1 \le i \le n, 1 \le j \le l_k, \\ &\quad j \equiv a_k (\mathrm{mod}\ 2), (r_i, t_k) \in E(G)\} \end{aligned}$$

Here, we assume that $a_k = 0$ if it is even, and $a_k = 1$ if it is odd. The weight of each arc (x,y) in D_k equals $w(x)$. Clearly, any path from s_k to t_k in D_k visits at most l_k buffers, and the number of visited buffers has the parity prescribed by a_k. Moreover,

Lemma 2 *The shortest path from s_k to t_k in D_k contains at most one edge of the form $(r_{i,j}, r_{i,j+1})$.*

Proof. Suppose it contains two such edges. Then, by "sliding-up" the path along the two edges we obtain a shorter path that visits two fewer buffers and hence still has the right parity. □

Notice also that, since D_k is acyclic, the shortest path connecting s_k to t_k can be computed in $O(|E(D_k)|)$ time.[6]

4 Rounding of Fractional MCF

In the previous section we presented an algorithm for approximating the optimum fractional multicommodity flow within any desired accuracy. Since the total flow f_t provides an upper bound on the optimum integral multi-commodity flow, we will be able to connect no more than f_t source-sink pairs under the given capacity, parity, and path-length constraints. In this section we show how to use the fractional flow to obtain a valid routing that contains almost f_t paths. The construction is based on the *randomized rounding* technique of Raghavan and Thomson [21] (see also [19]). For $k = 1, \ldots, K$, let $f_k(e)$ denote the flow of commodity k routed along arc $e \in E(D_k)$, $f_k(u)$ denote the flow of commodity k routed through vertex $u \in V(D_k)$, and $f_k = f_k(s_k)$ denote the total flow of commodity k. Since the underlying graph is in our case directed and acyclic, randomized rounding can be efficiently implemented as a random walk; see Algorithm 2.

Algorithm 2: Randomized MCF rounding

Input: Flows $f_k(e) \in [0,1]$, $k = 1, \ldots, K$, $e \in E(G)$ satisfying capacity constraints
Output: Set P of paths connecting pairs (s_k, t_k)

1. For each $k = 1, \ldots, K$, with probability f_k, do
   // Find path from s_k to t_k using a random walk based on $f_k(e)$'s
   $u \leftarrow s_k$
   While $u \neq t_k$ do
      Pick arc (u,v) with probability $f_k(u,v)/f_k(u)$
      $u \leftarrow v$

Note that the set P of paths of the given graph G produced by the randomized MCF rounding algorithm contains a path connecting the k^{th} source-sink pair with probability f_k. Hence, P contains on average $\sum_{i=1}^{K} f_k = f_t$ source-sink pairs. It is easy to see that the probability that node v is visited during the random walk for commodity k is equal to $f_k(v)$. As suggested in [19] for the edge-capacitated case, ensuring that no node capacities are exceeded can be accomplished by solving a multicommodity flow problem with capacities scaled down by a factor of $1-\varepsilon$ for a sufficiently small ε. Using Chernoff bounds on the sum of independent Bernoulli trials [21, 19] we can prove that, with high probability, this method routes an almost optimum number of nets without violating any node capacity:

Theorem 3 *If $c(v) \ge 5.2\ln(4n)$ for every vertex v in the given graph G, then, for any positive $\varepsilon < (\sqrt{5}-1)/2$, $|P| \ge (1-\varepsilon)^2 OPT$ with probability of at least $1 - \frac{1}{n} - 2e^{-0.38\varepsilon^2 OPT}$, where OPT is the optimum number of routable nets.*

In our experiments we have found that instead of scaling down node capacities, it is preferable to use a simpler approach: repeatedly drop the path in P that visits the most over-used nodes, until feasibility is achieved. We will call this the *greedy-deletion algorithm*.

5 Implemented Algorithms

In this section we will describe all implemented algorithms for the Global Buffering Problem. We first give a naive greedy algorithm and then describe a full implemented version of the solution based on the rounded approximate solution of the corresponding multicommodity flow. Finally, we describe a so-called *1-Shot heuristic* which tries to find an integer solution avoiding fractional relaxation.

Greedy Algorithm. If we just connect all source-sink pairs with shortest paths, then we may greatly overuse the buffer resource and obtain a non-feasible solution. If we make it feasible by using the "negative" greedy algorithm from the previous section, some pairs may still be routable afterward. We suggest a simpler algorithm which obtains a maximal feasible solution, i.e., such that no disconnected pairs can be routed. The running time is $O(KE)$ where K is the number of source-sink pairs and E is the number of edges in the graph G.

Algorithm 3: Greedy Routing Algorithm

Input: Graph G with the source-sink pairs (s_i, t_i)
Output: Set of paths connecting pairs (s_k, t_k)

1. For each sink-source pair (s_i, t_i) do
   if there exists an $(s_i - t_i)$-path satisfying parity and length constraints
   then find the shortest such path P and for each buffer block r on P decrease the capacity of r by 1 unit
   if resulting capacity of r is 0, then delete r from the graph G

ε-MCFG Algorithm. The rounded ε-approximate solution for the multi-commodity flow formulation will give us a feasible solution for the Global Buffering Problem. Since the fractional relaxation is solved only approximately, the rounded solution may be not maximal, i.e., some extra pairs can be routed. Experimental results below show that the Greedy Algorithm applied on top of the rounded solution can significantly increase the number of routed nets.

The 1-Shot Heuristic. The approximation algorithm for fractional multi-commodity flow uses a very simple increment of buffer block weights used in the shortest (s_i, t_i)-path. Intuitively, the weight increment forces subsequently routed nets to avoid usage of these buffer blocks. The idea of the 1-Shot heuristic is to apply that approach directly to the integer flow formulation. The resulting flow

[6] We have also considered the variant formulation where an overall cost bound B is given, along with a weight b_k for each source-destination pair (s_k, t_k). Define the cost of a route to be "the number of used buffer sites, times the weight of the source-destination pair". Then, we would like to solve the problem of maximizing $\sum_{p \in P} f_p$, subject to the additional constraint that the total cost of all (fractional) routes is not more than B. A variant of Algorithm 1 provably achieves a $(1+8\varepsilon)$-approximation for this **Cost-Constrained MFRVBB** problem; we omit the details of this algorithm due to space constraints.

Algorithm 4: ε-MCF Algorithm with Greedy Enhancement (ε-MCFG)

Input: Graph G with the source-sink pairs (s_i,t_i)
Output: Set of paths connecting pairs (s_k,t_k)

1. Solve the Fractional MCF Problem with ε-approximation Fleischer's algorithm (Algorithm 1)
2. Round the approximate fractional solution using random walk algorithm (Algorithm 2)
3. Using the greedy-deletion algorithm find feasible integer solution of Global Buffering Problem
4. Using Algorithm 3, augment the solution from step (3) to obtain a maximal feasible solution.

may be infeasible due to overuse of some buffer blocks, therefore we use the greedy-deletion algorithm to make it feasible. Finally we apply the same greedy enhancement that we use in ε-MCFG (Algorithm 4).

Algorithm 5: 1-Shot Heuristic

Input: Graph G with the source-sink pairs (s_i,t_i)
Output: Set of paths connecting pairs (s_k,t_k)

1. Assign weight 1 to each buffer block in the graph G.
2. While total overused capacity does not go down for 10 iteration do

   For each i, find a shortest (s_i,t_i)-path P_i in G

   For each buffer block r,

   set $w(r) = w(r) \cdot (1 + c_u(r)/c(r))$
3. Using greedy-deletion algorithm find feasible solution of Global Buffering Problem
4. Using Algorithm 3, augment the solution from step (3) to obtain a maximal feasible solution.

6 Implementation Experience

6.1 Experimental setup

All experiments were conducted on an SGI Origin 2000 with 16 195MHz MIPS R10000 processors (only one of which is actually used by the sequential implementations included in our comparison) and 4GB of internal memory, running under IRIX 6.4 IP27. Timing (reported in CPU seconds) was performed using low-level Unix interval timers, under similar load conditions for all experiments. All algorithms were coded in C and compiled using `gcc` version egcs-2.90.27 with `-O4` optimization.

The test instances used in our experiments were extracted from the next-generation microprocessor chip at SGI. We used an optimized floorplan of the circuit blocks and also optimized the location of the source/sink pin locations based on coarse timing budgets. We used $U = 4000\mu$m, and varied L between 500μm and 2000μm (for $L = 500\mu$m the design is typically gate dominated, for $L = 2000\mu$m the design tends to be totally wire dominated). Path-length upper bounds were computed with the formula $l_i = \text{dist}(s_k,t_k)/1000$. In our experiments, we used two different block capacities: 400, respectively 50, buffers per block. The latter value heavily constrains the routing – only a little over half of the nets can be routed – while the former usually allows routing of all nets.

At very early stages of chip planning, where our work applies, it is reasonable to consider a single repeater size (typically between 60x and 80x times minimum inverter, to give good energy-delay product as well as global line delay [2]). This is reasonable since delay is actually not very sensitive to either repeater size or repeater location (within reasonable bounds), and since we are considering very long global wires. Of course, source resistance and sink input capacitance can be significantly different from those of the standard repeater: to address this, shorter distances between source and the first repeater, or longer distances between the sink and the last repeater, can be enforced by constructing a restricted set of graph G edges incident to sources and sinks.

In all instances considered the number of nets was large (around 4000), and the number of buffer blocks small (50); such values are typical for this application. Our implementation attempted to exploit this particular structure of the problem for achieving practical MCF running times. To our knowledge, MCF instances of this size have never been solved before, neither exactly, using linear programming methods [1, 4, 15], nor approximately [17, 10]. A key speed-up idea was to keep individual directed acyclic graphs for each source-sink pair. This allows shortest path computations to be performed on graphs with no useless arcs (i.e., arcs leading into different sinks). Although this representation introduces some redundancy – buffer blocks and the arcs between them are now represented K times – this representation still fits comfortably in the internal memory of a workstation.

6.2 Results

Table 1 shows the number of routed nets and the running time required by each of the algorithms included in our comparison. Note that the running time of ε-MCFR grows quadratically in $1/\varepsilon$. As proved in Section 3 and illustrated in Figure 1, the quality of the fractional MCF depends linearly on ε, and this clearly affects the quality of the integral solution obtained by rounding. Quite surprisingly, however, this does not necessarily mean that we always need to solve the fractional flow with very good precision. As shown in Figure 1, running the Greedy routing algorithm starting from the solution obtained by randomized rounding leads to almost the same quality solutions for all values of ε.

It can be noted that the 1-Shot heuristic, which is only slightly slower than the Greedy routing, gives significantly improved solutions. On lightly constrained instances (block capacity of 400) the 1-Shot heuristic almost matches the performance of $\varepsilon - MCFG$. However, on highly constrained instances $\varepsilon - MCFG$ retains an advantage over the 1-Shot heuristic, especially for small values of ε.

7 Conclusions and Future Directions

In this paper, we addressed the problem of how to perform buffering of global nets *given an existing buffer block plan*. Assuming as in [6, 25] that global nets have been already decomposed into two-pin connections, we gave a provably good algorithm based on a recent approach of Garg and Könemann [8] (see also [7]). Our method routes connections using available buffer blocks, such that required upper and lower bounds on buffer intervals – as well as wirelength upper bounds per connection – are satisfied. Unlike [6, 25], our model allows more than one buffer to be inserted into any given connection. In addition, our algorithm observes buffer parity constraints, i.e., it will choose to use an inverter or a buffer (= co-located pair of inverters) according to source and destination signal parity. The algorithm outperforms an implementation of "greedy ripup and reroute", and has been validated on top-level layouts extracted from a recent high-end microprocessor design. It gives very good results on real-world test cases.

Our current research pursues several extensions of the MCF based approach. Examples include the following. (1) We incorporate

ID	L	U	C	K	Greed	1-Shot	ε-MCFG				
							$\varepsilon = 0.16$	$\varepsilon = 0.08$	$\varepsilon = 0.04$	$\varepsilon = 0.02$	$\varepsilon = 0.01$
i1	2000	4000	400	3962	3883 / 19.86	3960 / 33.92	3962 / 109.26	3962 / 292.48	3962 / 1118.79	3962 / 4412.43	3962 / 16239.83
i2	1000	4000	400	4191	4070 / 27.34	4186 / 38.28	4191 / 109.59	4191 / 345.35	4191 / 1298.24	4191 / 4912.63	4191 / 18835.71
i3	500	4000	400	4212	4101 / 28.79	4207 / 39.62	4212 / 107.09	4212 / 334.17	4212 / 1211.52	4212 / 4639.23	4212 / 17929.95
i4	2000	4000	50	3962	2148 / 19.77	2179 / 23.21	2325 / 074.55	2334 / 233.80	2339 / 850.85	2347 / 3293.29	2340 / 13106.98
i5	1000	4000	50	4191	2216 / 27.32	2236 / 34.23	2378 / 109.75	2369 / 295.99	2393 / 916.63	2394 / 3659.62	2392 / 14523.98
i6	500	4000	50	4212	2223 / 28.81	2232 / 36.14	2378 / 093.26	2382 / 274.63	2389 / 916.61	2392 / 3627.42	2394 / 15872.07

Table 1: Number of routed nets / running time for the implemented algorithms

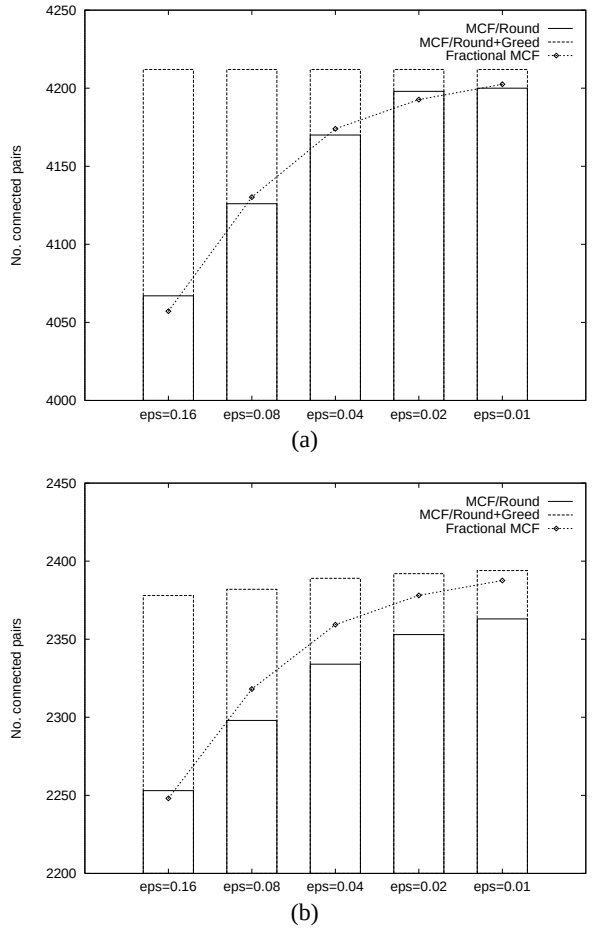

Figure 1: The benefit of running Greedy on top of rounded MCF on instances i3 (a) and i6 (b)

more precise cost models for usage of buffer blocks. For example, in a regime where power routing to buffer blocks is expensive, the first time a given buffer block is used should incur *fixed* cost, while second and subsequent uses (up to capacity of the station) each incur *marginal* cost. (2) We would like to group source-sink pairs that have a common source, so as to achieve Steiner routing solutions for multi-pin nets. On each source-sink path, the repeater interval constraint $[L,U]$ would still hold. (3) With N-layer metal processes, different wiring "tiers" (layer-pairs) have different performance attributes. Thus, for example, the repeater interval would be between 2500-3000μm on M3-M4, and between 4000-5000μm on M7-M8. We would like to solve the Global Buffering Problem when several routing tiers are available (each with a certain capacity for pin-to-buffer and buffer-to-buffer connections), and each connection must be routed entirely on a single tier. (4) Finally, we believe that improved solutions to the original buffer block planning (placement) problem are still possible.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.L. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] A. E. Caldwell, Y. Cao, A. B. Kahng, F. Koushanfar, H. Lu, I. L. Markov, M. R. Oliver, D. Stroobandt and D. Sylvester, "GTX: The MARCO GSRC Technology Extrapolation System", *Proc. DAC*, 2000, pp. 693–698.

[3] R. C. Carden and C.-K. Cheng, "A Global Router Using an Efficient Approximate Multicommodity Multiterminal Flow Algorithm", *Proc. DAC*, 1991, pp. 316–321.

[4] J. Castro and N. Nabona, "An Implementation of Linear and Non-Linear Multicommodity Network Flows", *Eur. J. Oper. Res.* 92 (1996), pp. 37–53.

[5] J. Cong, L. He, C.-K. Koh and P. H. Madden, "Performance Optimization of VLSI Interconnect Layout", *Integration* 21 (1996), pp. 1–94.

[6] J. Cong, T. Kong and D. Z. Pan, "Buffer Block Planning for Interconnect-Driven Floorplanning", *Proc. ICCAD*, 1999, pp. 358–363.

[7] L. K. Fleischer, "Approximating Fractional Multicommodity Flow Independent of the Number of Commodities", *Proc. 40th Annual Symposium on Foundations of Computer Science*, 1999, pp. 24–31.

[8] N. Garg and J. Könemann, "Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems", *Proc. 39th Annual Symposium on Foundations of Computer Science*, 1998, pp. 300–309.

[9] R. Goering, "Cadence looks to overhaul chip design flow", *EE Times*, May 15, 1999.

[10] A.V. Goldberg, J.D. Oldham, S. Plotkin, and C. Stein, "An Implementation of a Combinatorial Approximation Algorithm for Minimum-Cost Multicommodity Flow", *Proc. Integer Programming and Combinatorial Optimization*, LNCS 1412, Springer, Berlin, 1998, pp. 338–352.

[11] J. Huang, X.-L. Hong, C.-K. Cheng and E. S. Kuh, "An Efficient Timing-Driven Global Routing Algorithm", *Proc. DAC*, 1993, pp. 596–600.

[12] A. B. Kahng, S. Muddu and E. Sarto, "Tuning Strategies for Global Interconnects in High-Performance Deep-Submicron ICs", *VLSI Design* 10(1) (1999), pp. 21–34.

[13] A. B. Kahng, S. Muddu and E. Sarto, "On Switch Factor Based Analysis of Coupled RC Interconnects", *Proc. DAC*, 2000, pp. 79–84.

[14] A. B. Kahng, S. Muddu, E. Sarto and R. Sharma, "Interconnect Tuning Strategies for High-Performance ICs", *Proc. DATE*, 1998.

[15] A. Kamath and O. Palmon, "Improved Interior Point Algorithms for Exact and Approximate Solution of Multicommodity Flow Problems", *Proc. 6th Annual ACM–SIAM Symp. on Discrete Algorithms*, 1995, pp. 502–511.

[16] M. Kang, W. Dai, T. Dillinger and D. LaPotin, "Delay Bounded Buffered Tree Construction for Timing Driven Floorplanning", *Proc. ICCAD*, 1997, pp. 707–712.

[17] T. Leong, P. Shor, and C. Stein, "Implementation of a Combinatorial Multicommodity Flow Algorithm", In D.S. Johnson and C.C. McGeoch, eds., *Network Flows and Matching*, vol. 12 of *Series in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, 1993, pp. 387–405.

[18] J. Lillis, C. K. Cheng and T. T. Y. Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model", *Proc. ICCAD*, 1995, pp. 138–143.

[19] R. Motwani, J. Naor, and P. Raghavan, "Randomized Approximation Algorithms in Combinatorial Optimization", In *Approximation Algorithms for NP-hard Problems* (Boston, MA, 1997), D. Hochbaum, ed., PWS Publishing, pp. 144–191.

[20] T. Okamoto and J. Cong, "Buffered Steiner Tree Construction With Wire Sizing for Interconnect Layout Optimization", *Proc. ICCAD*, 1996, pp. 44–49.

[21] P. Raghavan and C.D. Thompson, "Randomized Rounding", *Combinatorica* 7 (1987), pp. 365–374.

[22] J. O'Rourke, *Computational Geometry in C*, Cambridge University Press, 1993.

[23] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, 1999.

[24] E. Shragowitz and S. Keel, "A Global Router Based on a Multicommodity Flow Model", *Integration* 5(1) (1987), pp. 3–16.

[25] X. Tang and D. F. Wong, "Planning Buffer Locations by Network Flows", *Proc. ISPD*, 2000, pp. 180–185.