# Floorplan Evaluation with Timing-Driven Global Wireplanning, Pin Assignment, and Buffer/Wire Sizing[*]

Christoph Albrecht,[†] Andrew B. Kahng, Ion Măndoiu, and Alexander Zelikovsky[‡]
CSE Department, UCSD, La Jolla, CA 92093-0114
[†]Research Institute for Discrete Mathematics, University of Bonn, Lennéstr. 2, 53113 Bonn, Germany
[‡]CS Department, Georgia State University, Atlanta, GA 30303
albrecht@or.uni-bonn.de, {abk,mandoiu}@cs.ucsd.edu, alexz@cs.gsu.edu

## Abstract

*We describe a new algorithm for floorplan evaluation using timing-driven buffered routing according to a prescribed buffer site map. Specifically, we describe a provably good multi-commodity flow based algorithm that finds a global routing minimizing routing area (wirelength and number of buffers) subject to given constraints on buffer/wire congestion and sink delays. This permits detailed floorplan evaluation, i.e., computing the tradeoff curve between routing area and wire/buffer congestion under any combination of delay and capacity constraints.*

*Our algorithm (1) enforces maximum source/buffer wireloads; (2) enforces wire and buffer congestion constraints by taking into account routing channel capacities and buffer site locations; (3) enforces individual sink delay constraints; (4) performs buffer/wire sizing and layer assignment; and (5) integrates pin assignment with virtually no increase in runtime. Preliminary experiments show that near-optimal results are obtained with a practical runtime.*

## 1. Introduction

Early planning of buffer and wiring resources is a critical aspect of every modern high-performance VLSI implementation methodology. Today, such planning is needed to evaluate the quality of RT-level partitioning and soft (pre-synthesis) block placement/shaping, system-level timing constraints, and pin definition and buffered routing of global interconnects.

The requirements for global wire planning as an adjunct to floorplan definition have not changed very much from those set out in pioneering works as BBL [6] and BEAR [10]: the floorplan definition must be aware of congestion, wirelength, timing, etc. Similarly, the need for simultaneous pin assignment and global routing has not changed since, e.g., [7]. It is well-understood that today's context for floorplan definition and global wire planning has evolved: (1) channel-less multilayer area routing has replaced channel/switchbox routing; (2) interconnect delays are more balanced with (appropriately sized) gate delays, and no longer dominated by gate delays; (3) layer RC constants vary by factors of up to 100×, so that layer assignment must be planned; (4) global interconnects are buffered; and (5) floorplanning is at the RT-level (instead of physical floorplanning) with soft blocks having uncertain area/delay envelopes. At the same time, the underlying problem formulations and algorithmic technologies have *separately* advanced in at least three important ways:

- The *buffer block* methodology, along with the associated planning problem (i.e., solving for locations and capacities of buffer blocks), was proposed by Cong et al. [9] and further elucidated by Tang and Wong [18]. While the buffer block methodology has indeed been used throughout the late 1990s in hierarchical structured-custom (high-end microprocessor) methodologies, it may be less relevant to flat or ASIC-like regimes due to issues of separate power distribution, congestion, etc. To alleviate congestion problems associated with the use of buffer blocks, the DAC'2001 work of Alpert et al. [3] proposed a *buffer site* methodology which more uniformly distributes buffers across the chip "wherever possible".
- The increased impact of interconnects on system performance in deep-submicron technologies has led to a large literature on performance-driven optimizations *for individual global nets*. Such optimizations include buffer insertion and sizing, wire sizing, and topology synthesis, as comprehensively surveyed in [8]. We note that for the purposes of buffered global wire planning, *it is likely suf-*

[*]Work supported in part by Cadence Design Systems, Inc., the MARCO Gigascale Silicon Research Center, and NSF Grant CCR-9988331.

*ficient that the tool be able to exploit the availability of multiple buffer types, multiple wire widths, and multiple layer assignments*.

- A literature on *provably good* global routing has been developed based on the primal-dual framework, starting with "column-generating" analogies in [4, 5], then continuing with the work of Albrecht [1] exploiting recent fast approximations for multi-commodity flows [15, 14]. The work of Dragan et al. [11, 12, 13] has even more recently applied such provable approximations to the problem of global routing with a *prescribed buffer block plan*, taking into account signal parity, delay upper/lower bounds, and other practical considerations.

Our ongoing work seeks to combine the above separate threads of the recent literature into a coherent approach to floorplan definition, global route planning, and pin assignment. In some sense, we hope to regain the "holistic" perspective of the earliest works cited above, e.g., BBL/BEAR. Eventually, we seek to overcome the individual limitations of various "component" technologies: (1) single-net performance optimizations should adapt to the congestion-aware global routing context; (2) the buffer block framework should adapt to the more continuous buffer site framework; (3) *both* congestion and timing should be addressed simultaneously (e.g., the work of Cong et al. [9] addresses timing but not congestion; the work of Alpert et al. [3] addresses congestion but not timing); (4) pin assignment (which is especially important in soft block planning) as well as buffer insertion/sizing, wire sizing, and layer assignment degrees of freedom should be exploited; and (5) bounds on performance ratio should be maintained as in "provably good" methods. In this paper, we present our initial efforts toward this end. Our contribution is summarized as follows.

- We describe a new algorithm for floorplan evaluation using timing-driven buffered routing according to a prescribed buffer site map. Specifically, we describe a provably good multi-commodity flow based algorithm that finds a global routing minimizing routing area (wirelength and number of buffers) subject to given constraints on buffer/wire congestion and sink delays. Following Alpert et al. [3], we use a 2-dimensional tile graph to capture the wire and buffer congestion of a given floorplan. The tile size depends on the desired tradeoff between estimation accuracy and runtime.
- Our implementation permits detailed floorplan evaluation in that it enables computing the tradeoff curve between routing area and wire/buffer congestion under any combination of delay and capacity constraints.
- Like the allocation heuristic in [3], our algorithm enforces maximum source/buffer wireloads and controls congestion by taking into account routing channel capacities and buffer site locations. At the same time, like the buffer-block planning algorithm in [9], our algorithm takes into account individual sink delay constraints.
- Simultaneously, our algorithm performs buffer and wire sizing by taking into account given libraries of buffer types and wire widths, and integrates layer and pin assignment (the latter with virtually no increase in runtime). Soft pin locations are modeled as multiple sites (grid locations), and are enabling to solution quality.

The paper is organized as follows. In Section 2 we formalize the floorplan evaluation problem for 2-pin nets. In Section 3 we reformulate the problem as a minimum cost integer multicommodity flow problem (with capacities on *sets of edges*), give an efficient algorithm for finding near-optimal solutions to the fractional relaxation, and show how to convert fractional solutions to near-optimal routings by randomized rounding. In Section 4 we give the modifications needed to enforce sink delay constraints and take into account simultaneous buffer and wire sizing and/or layer assignment. We discuss handling of multi-pin nets in Section 5 and conclude the paper with experimental results comparing our algorithm with the allocation heuristic in [3].

## 2. Problem formulation

For a given floorplan and tile size, we construct a vertex- and edge-weighted *tile graph* $G = (V, E, b, w)$, where

- $V$ is the set of tiles;
- $E$ contains an edge between any two adjacent tiles;
- For each tile $v \in V$, the *buffer capacity* $b(v)$ is the number of buffer sites located in $v$; and
- For each edge $e = (u, v) \in E$, the *wire capacity* $w(e)$ is the number of routing channels available between tiles $u$ and $v$.

To simplify the presentation, in this and the following two sections we will assume that all nets have 2 pins; multi-pin nets are considered in Section 5. We denote by $\mathcal{N} = \{N_1, N_2, \ldots, N_k\}$ the given netlist, where each net $N_i$ is specified by the sets of tiles, $S_i \subseteq V$ and $T_i \subseteq V$, to which the source, respectively the sink of $N_i$ can be assigned.

We consider first the case when a single buffer and wire width are available, and only buffer wireload constraints must be satisfied. In this case we seek for each net $N_i$ a path $P_i$ buffered using the available buffer sites and connecting a vertex from $S_i$ to a vertex from $T_i$ (see Figure 1) such that the source vertex and the buffers drive each at most $U$ units of wire, where $U$ is a given upper-bound (the example in Figure 1 has $U = 5$). Formally, a *feasible buffered routing* for net $N_i$ is a path $P_i = (v_0, v_1, \ldots, v_{l_i})$ in $G$ together with a set of buffers $B_i \subseteq \{v_0, \ldots, v_{l_i}\}$ such that:

- $v_0 \in S_i$ and $v_{l_i} \in T_i$;
- $w(v_{i-1}, v_i) \geq 1$ for every $i = 1, \ldots, l_i$;
- $b(v_i) \geq 1$ for every $v_i \in B_i$; and

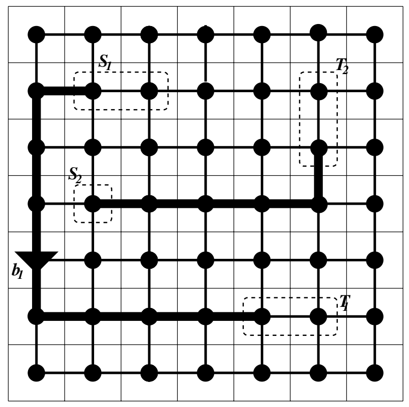

**Figure 1. Tile graph with two 2-pin nets.**

- The length along $P_i$ between $v_0$ and the first buffer in $B_i$, between consecutive buffers, and between the last buffer and $v_{l_i}$, are all at most $U$.

We will denote by $\mathcal{R}_i$ the set of all feasible routings $(P_i, B_i)$ for net $N_i$. Given buffered routings $(P_i, B_i) \in \mathcal{R}_i$ for each net $N_i$, the relative *buffer congestion* is

$$\mu = \max_{v \in V} \frac{|\{i : v \in B_i\}|}{b(v)}$$

and the relative *wire congestion* is

$$\nu = \max_{e \in E} \frac{|\{i : e \in P_i\}|}{w(e)}$$

The buffered paths $(P_i, B_i)$, $i = 1, \ldots, k$, are simultaneously routable iff both $\mu \leq 1$ and $\nu \leq 1$. To leave resources available for subsequent optimization of critical nets and ECO routing, we will generally seek simultaneous buffered routings with buffer and wire congestion bounded away from 1. Using the total wire and buffer area as measure of floorplan quality we get:

**Floorplan Evaluation Problem (FEP)**
**Given:**

- Grid-graph $G = (V, E, b, w)$, with buffer capacities $b(v)$ and wire capacities $w(e)$;
- Set $\mathcal{N} = \{N_1, \ldots, N_k\}$ of 2-pin nets with unassigned source and sink pins $S_i, T_i \subseteq V$; and
- Wireload, buffer congestion, and wire congestion upper-bounds $U > 0$, $\mu_0 \leq 1$, and $\nu_0 \leq 1$.

**Find:** feasible buffered routings $(P_i, B_i) \in \mathcal{R}_i$ for each net $N_i$ with relative buffer congestion $\mu \leq \mu_0$ and relative wire congestion $\nu \leq \nu_0$, minimizing the total wire and buffer area, i.e., $\alpha \sum_{i=1}^{k} |B_i| + \beta \sum_{i=1}^{k} |P_i|$, where $\alpha, \beta \geq 0$ are given constants.

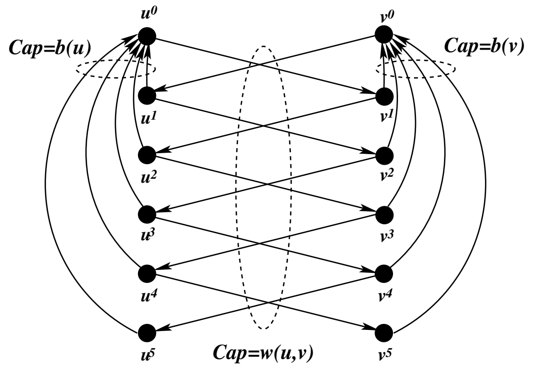

**Figure 2. Gadget replacing edge $(u, v)$ for $U = 5$.**

## 3. Solving FEP via multicommodity flow approximation

Recall that, for every feasible buffered routing in the tile graph $G = (V(G), E(G), b, w)$, the wireload of the source and of each buffer must be at most $U$. We start by defining a directed graph $H$ which captures exactly these feasible buffered routings. The graph $H$ has $U + 1$ vertices $v^0, v^1, \ldots, v^U$ for each vertex $v \in V(G)$. Every edge $(u, v) \in E(G)$ is replaced in $H$ by directed arcs of the form $(u^{i-1}, v^i)$ and $(v^{i-1}, u^i)$. In addition, "buffer" arcs $(v^i, v^0)$, $i = 1, \ldots, U$, are added to $H$ for every vertex $v \in V(G)$ with $b(v) \geq 1$ (see Figure 2). Finally, vertices corresponding to each net source and sink are added to $H$ and connected to the tiles to which they may be assigned. Formally, the graph $H$ has vertex set

$$V(H) = \{s_i, t_i \mid 1 \leq i \leq k\} \cup \{v^j \mid v \in V(G), 0 \leq j \leq U\}$$

and arc set

$$E(H) = E_{src} \cup E_{sink} \cup \Big( \bigcup_{(u,v) \in E(G)} E_{u,v} \Big) \cup \Big( \bigcup_{v \in V(G)} E_v \Big)$$

where

$$\begin{aligned} E_{src} &= \{(s_i, v^0) \mid v \in S_i, 1 \leq i \leq k\} \\ E_{sink} &= \{(v^j, t_i) \mid v \in T_i, 0 \leq j \leq U, 1 \leq i \leq k\} \\ E_{u,v} &= \{(u^{j-1}, v^j), (v^{j-1}, u^j) \mid 1 \leq j \leq U\} \\ E_v &= \{(v^j, v^0) \mid 1 \leq j \leq U\} \end{aligned}$$

Since every $s_i$–$t_i$ path in $H$ can visit at most $U$ vertices between any two buffer arcs, we get:

**Lemma 3.1** *There is a 1-to-1 correspondence between the feasible buffered routings for net $N_i$ in the tile graph $G$ and the $s_i$–$t_i$ paths in $H$.*

We will use the correspondence established in Lemma 3.1 to give an integer linear program (ILP) formulation for the floorplan evaluation problem. Let $\mathcal{P}_i$ denote the set of all simple $s_i$–$t_i$ paths in $H$. We introduce a $0/1$ variable $x_p$ for every path $p \in \mathcal{P} := \cup_1^k \mathcal{P}_i$. The variable $x_p$ is set to 1 if the buffered routing corresponding to $p \in P_i$ is used to connect net $N_i$, and to 0 otherwise. With this notation, FEP can be formulated as follows:

$$\min \sum_{p\in\mathcal{P}} \Big(\alpha \sum_{v\in V(G)} |p\cap E_v| + \beta \sum_{(u,v)\in E(G)} |p\cap E_{u,v}|\Big)\, x_p \quad (1)$$

subject to

$$\begin{array}{ll} \sum\limits_{p\in\mathcal{P}} |p\cap E_v|\, x_p \le \mu_0\, b(v), & v\in V(G) \\ \sum\limits_{p\in\mathcal{P}} |p\cap E_{u,v}|\, x_p \le \nu_0\, w(u,v), & (u,v)\in E(G) \\ \sum\limits_{p\in\mathcal{P}_i} x_p = 1, & i=1,\ldots,k \\ x_p \in \{0,1\}, & p\in\mathcal{P} \end{array}$$

Solving exactly ILP (1) is NP-hard. Our approach is to solve its fractional relaxation (obtained by replacing the constraints $x_p \in \{0,1\}$ with $x_p \ge 0$) and then obtain near-optimal integer solutions by randomized rounding.

The fractional relaxation of ILP (1) is a minimum cost multicommodity flow problem with capacity constraints on *sets of edges*; such capacity constraints have been previously considered in [13] for the maximum multicommodity flow problem. Instead of solving this relaxation directly, we introduce an upper bound $D$ on the wire and buffer area and consider the following linear program (LP):

$$\min \lambda \quad (2)$$

subject to

$$\begin{array}{ll} \sum\limits_{p\in\mathcal{P}} \Big(\alpha \sum\limits_{v\in V(G)} |p\cap E_v| + \beta \sum\limits_{(u,v)\in E(G)} |p\cap E_{u,v}|\Big)\, x_p \le \lambda\, D & \\ \sum\limits_{p\in\mathcal{P}} |p\cap E_v|\, x_p \le \lambda\, \mu_0\, b(v), & v\in V(G) \\ \sum\limits_{p\in\mathcal{P}} |p\cap E_{u,v}|\, x_p \le \lambda\, \nu_0\, w(u,v), & (u,v)\in E(G) \\ \sum\limits_{p\in\mathcal{P}_i} x_p = 1, & i=1,\ldots,k \\ x_p \ge 0, & p\in\mathcal{P} \end{array}$$

Let $\lambda^*$ be the optimum objective value for LP (2). Solving the fractional relaxation of ILP (1) is equivalent to finding the minimum $D$ for which $\lambda^* \le 1$. This can be done by a binary search which requires solving the LP (2) for each probed value of $D$; a similar approach was used in [15] for solving the minimum cost concurrent multicommodity flow problem. A lower bound on the optimal value of $D$ can be derived by ignoring all buffer and wire capacity constraints, i.e., by computing for each net $N_i$ buffered paths $p \in \mathcal{P}_i$ minimizing $\alpha \sum_{v\in V(G)} |p\cap E_v| + \beta \sum_{(u,v)\in E(G)} |p\cap E_{u,v}|$. A trivial upper bound is the total routing area available, i.e., $\alpha\, \mu_0 \sum_{v\in V(G)} b(v) + \beta\, \nu_0 \sum_{(u,v)\in E(G)} w(u,v)$.

The algorithm for approximating the optimum solution to LP (2) (see Figure 3) uses the general framework for multicommodity flow approximation introduced in [15], and relies on simultaneously approximating the *dual* LP:

$$\max \sum_{i=1}^{k} l_i \quad (3)$$

subject to

$$\begin{array}{l} \sum\limits_{v\in V(G)} \mu_0 b(v) y_v + \sum\limits_{(u,v)\in E(G)} \nu_0 w(u,v) z_{u,v} + Du = 1 \\ \sum\limits_{v\in V(G)} |p\cap E_v|\,(y_v+\alpha u) \\ \qquad + \sum\limits_{(u,v)\in E(G)} |p\cap E_{u,v}|\,(z_{u,v}+\beta u) \ge l_i,\ p\in\mathcal{P}_i \\ y_v \ge 0, \quad v\in V(G) \\ z_e \ge 0, \quad e\in E(G) \end{array}$$

The algorithm starts with trivial solutions for LPs (2) and (3), then updates these solutions over several phases. In each phase (lines 5–15) one unit of flow is routed for each commodity; a feasible solution is obtained in the end after dividing all flows by the number of phases. Commodities are routed along paths with minimum weight w.r.t. weights of $y_v + \alpha u$ for arcs in $E_v$, $v \in V(G)$, of $z_{u,v} + \beta u$ for arcs in $E_{u,v}$, $(u,v) \in E(G)$, and of 0 for all the other arcs (cf. LP (3)). The dual variables are increased by a multiplicative factor for all vertices/edges on a routed path; this ensures that dual weights increase exponentially with usage and thus often used edges are subsequently avoided.

Minimum-weight paths are computed using Dijkstra's single-source shortest path algorithm. To reduce the number of shortest path computations, paths are recomputed only when their weight increases by a factor of more than $(1+\gamma\varepsilon)$. This speed-up idea, first applied in [14] for the maximum multicommodity flow problem, has been shown in [1] to decrease the running time in practice while maintaining the same theoretical worst-case runtime.

**Theorem 3.2** *The algorithm in Figure 3 finds an* $(1+\varepsilon_0)$-*approximation with* $O\big(\frac{1}{\varepsilon_0^2 \lambda^*} k \ln n\big)$ *shortest path computations, using* $\varepsilon = \min\big\{\frac{1}{\gamma}, \frac{1}{\gamma}(\sqrt{1+\varepsilon_0}-1), \frac{1}{4}\big(1-\big(\frac{1}{1+\varepsilon_0}\big)^{1/6}\big)\big\}$ *and* $\delta = \big(\frac{1-\varepsilon'}{n+m}\big)^{1/\varepsilon}$, *where* $n$ *and* $m$ *are the number of vertices and edges of* $G$, *and* $\varepsilon' := \varepsilon(1+\varepsilon)(1+\varepsilon\gamma)$.

After solving LP (2) we route each net $N_i$ by randomly choosing one of the paths $p \in \mathcal{P}_i$ with probabilities given by the flows $x_p$. Randomized rounding guarantees that (for large enough capacities) the relative congestion increases only by a small amount. For details see [17].

**Remark.** Using ideas from [1] it can be shown that the algorithm in Figure 3 does not only minimize $\lambda$, but also "strives" for a lexicographically minimum solution w.r.t.

```
(1)  Set $y_v := \frac{\delta}{\mu_0 b(v)}\ \forall v \in V(G), \quad z_e := \frac{\delta}{\nu_0 w(e)}\ \forall e \in E(G), \quad u := \frac{\delta}{D}$
(2)  Set $x_p := 0\ \forall p \in \mathcal{P}$.
(3)  Set $p_i := \emptyset$ for $i = 1, ..., k$.
(4)  While $\mu_0 \sum_{v \in V(G)} b(v) y_v + \nu_0 \sum_{(u,v) \in E(G)} w(u,v) z_{u,v} + Du < 1$ do:
(5)  begin
(6)    For $i := 1$ to $k$, do
(7)    begin
(8)      If $p_i = \emptyset$ or $\sum_{v \in V(G)} |p_i \cap E_v| (y_v + \alpha u) + \sum_{(u,v) \in E(G)} |p_i \cap E_{u,v}| (z_{u,v} + \beta u) > (1 + \gamma\varepsilon) l_i$ then
(9)      begin
(10)       Find a path $p_i \in \mathcal{P}_i$ minimizing $l_i := \sum_{v \in V(G)} |p_i \cap E_v| (y_v + \alpha u) + \sum_{(u,v) \in E(G)} |p_i \cap E_{u,v}| (z_{u,v} + \beta u)$
(11)     end
(12)     Set $x_{p_i} := x_{p_i} + 1$
(13)     Set $y_v := y_v \left(1 + \varepsilon \frac{|p_i \cap E_v|}{\mu_0 b(v)}\right) \forall v \in V(G), \quad z_e := z_e \left(1 + \varepsilon \frac{|p_i \cap E_{u,v}|}{\nu_0 w(u,v)}\right) \forall (u,v) \in E(G)$
           $u := u \left(1 + \varepsilon \frac{\alpha \sum_{v \in V(G)} |p_i \cap E_v| + \beta \sum_{(u,v) \in E(G)} |p_i \cap E_{u,v}|}{D}\right)$
(14)   end
(15) end
```

**Figure 3. Approximation algorithm for LP (2) and its dual.**

the vector consisting of the relative buffer congestion of the vertices, the relative wire congestion of the edges, and the ratio between the total routing area and the upperbound $D$. Therefore, a solution of the algorithm indicates where possible changes to the floorplan have to be made in order to reach a feasible routing of all nets. For this it is especially useful to run the algorithm with a large value for $D$, that is to relax the constraint on the total area. If we want to ignore this constraint completely (i.e., set $D = \infty$), the dual variable $u$ is 0 during the whole algorithm and can be eliminated.

## 4. Timing-driven floorplan evaluation

In this section we address the floorplan evaluation problem under given sink delay constraints. First, we consider enforcing sink delay constraints when a single buffer type and wire width are available. Then, we extend our algorithm to simultaneously handle buffer and wire sizing.

### 4.1. Enforcing sink delay constraints

Assume that we are given an upper-bound of $d_i$ on the source-to-sink delay of net $N_i$. The Elmore delay of a buffered path is the sum of delays of the wire segments comprising the path. The delay of a wire segment connecting the source or buffer $u$ to the sink or buffer $v$ is the sum between the *gate delay*

$$intrinsic\_delay_u + r_u \cdot (c_w l_{u,v} + C_{in}(v))$$

and the *wire delay*

$$r_w l_{u,v} \cdot (c_w l_{u,v}/2 + C_{in}(v))$$

Here, $r_u$ and $C_{in}(u)$ are the output resistance, respectively input capacitance, of the buffer/terminal $u$, $r_w$ and $c_w$ are the resistance, respectively capacitance, of a tile-long wire, and $l_{u,v}$ is the wirelength (in tiles) between $u$ and $v$.

To simplify the exposition we will assume that the intrinsic delay and output resistance of sources are equal to the corresponding parameters of a buffer; in Section 4.2 we will give a construction that handles non-uniform source parameters. Under this assumption the total (i.e., gate + wire) delay of each routing segment depends only on the segment's length, $l$, and the input capacitance of the driven buffer/sink. Note that every routing segment ending in tile $v$ corresponds in $H$ to a path whose last arc is either the buffer arc $(v^l, v^0)$ if the segment drives a buffer, or the arc $(v^l, t_i)$ if the segment drives the $i$th sink. Since these arcs fully identify both the segment length and the input capacitance of the driven buffer/sink, we can assign them pre-computed segment delays, thus obtaining:

**Lemma 4.1** *The 1-to-1 correspondence between feasible buffered routings of net $N_i$ in $G$ and the $s_i$–$t_i$ paths in $H$ preserves the delay.*

To enforce the delay upper-bounds for the solution computed by the algorithm given in Section 3 we must restrict the computation to those paths $p \in \mathcal{P}_i$ that have delay at most $d_i$. Although the problem of finding a least weight

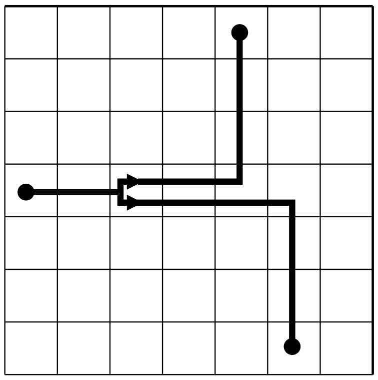

**Figure 4. Routing which is not a tree in $G$.**

$s_i$–$t_i$ path with bounded delay is NP-hard, there is a practical algorithm with arbitrarily good approximation guarantee, i.e., fully polynomial approximation scheme (FPTAS) [16]. This suffices for obtaining arbitrarily good approximations to the timing-driven floorplan evaluation problem:

**Theorem 4.2** *There is a fully polynomial approximation scheme for the timing-driven floorplan evaluation problem.*

## 4.2. Buffer and wire sizing

In this section we show how to take into account buffer and wire sizing during timing-driven floorplan evaluation. Consider first buffering with a given buffer library $\mathcal{B}$ and a fixed wire width. Each buffer type in $\mathcal{B}$ has a given area, input capacitance, and output resistance; the different buffer parameters also translate into different upperbounds $U$ on the wirelength that can be driven by a buffer of each type. The directed graph $H^{\mathcal{B}}$ capturing all feasible routings with buffers from $\mathcal{B}$ is obtained as follows. First construct, as in Section 4.1, a delay weighted graph $H$ for each buffer type, with the same source and sink nodes, $s_i$ and $t_i$, $i = 1, \ldots, k$, and also with the same nodes $v^0$ for every $v \in V(G)$. Then, remove all arcs of the form $(s_i, v^0)$, except the arc from the graph corresponding to the buffer whose driving strength is closest to that of $s_i$.

To reduce the complexity of wire sizing, we require a fixed wire width along any buffer-to-buffer wire segment. As shown in [2], this requirement may increase propagation delays by at most 5% compared to the optimum delay achieved by wire tapering. Simultaneous wire sizing can now be taken into account by a construction similar to the one used above for buffer sizing. Let $\mathcal{W}$ be a given library of wire widths. The directed graph $H^{\mathcal{B}\times\mathcal{W}}$ capturing all feasible buffered routings with buffers from $\mathcal{B}$ and wire widths from $\mathcal{W}$ is obtained by constructing for each wire width in $\mathcal{W}$ a graph $H^{\mathcal{B}}$ as above, with the same source and sink nodes, $s_i$ and $t_i$, $i = 1, \ldots, k$, and also with the same nodes $v^0$ for every $v \in V(G)$.

```
(1)  Set $w^* := \infty$
(2)  For all $v \in V$ do        // try all possible Steiner points
(3)  begin
(4)    For $j := 0$ to $U$
(5)    begin
(6)      Find a shortest $v^{U-j} - t_i^1$–path $P_1$ in $H$
(7)      For $k := 0$ to $U - j$
(8)      begin
(9)        Find a shortest $v^{U-k} - t_i^2$–path $P_2$ in $H$
(10)       Find a shortest $s_i^0 - v^{U-j-k}$–path $P_0$ in $H$
(11)       If $w(P_0) + w(P_1) + w(P_2) \leq w^*$ then
(12)         Set $w^* := w(P_0) + w(P_1) + w(P_2)$
                 $T^* := P_0 \cup P_1 \cup P_2$
(13)     end
(14)   end
(15) end
(16) return $T^*$
```

**Figure 5. Algorithm for finding minimum weight buffered routings for 3-pin nets.**

**Lemma 4.3** *There is a delay-preserving 1-to-1 correspondence between buffer- and wire-sized feasible routings of net $N_i$ in $G$ and the $s_i$–$t_i$ paths in $H^{\mathcal{B}\times\mathcal{W}}$.*

**Theorem 4.4** *There is a fully polynomial approximation scheme for the timing-driven floorplan evaluation problem with given buffer and wire width libraries.*

## 5. Solving FEP for multipin nets

For a multipin net we are looking for a buffered tree (instead of a buffered path) in which the wireload of each buffer is at most $U$. Such a buffered tree does not necessarily have to be a tree in the graph $G$. Figure 4 shows such an example: When $U = 6$ we need at least two buffers for any tree connecting the three terminals, and the cheapest (possibly the only available) place for these two buffers might be exactly the tile shown.

Extending the approximation algorithm from Section 3 to multipin nets requires a subroutine for computing feasible routings having minimum weight with respect to the dual variables. Figure 5 gives such a subroutine for 3-pin nets. We assume here that the possible locations of the source pin for a net $N_i$ are specified by $S_i$ as before, while the two sinks are specified by sets $T_i^1$ and $T_i^2$. In the graph $H$ we have vertices $t_i^1$ and $t_i^2$ and edge sets $\{(v^j, t_i^l) \mid v \in T_i^l, j = 0, ..., U\}$, $l = 1, 2$ for the sink pins of such a 3-pin net. For each possible Steiner point $v$, the algorithm tries all possible lengths $j$ and $k$ to the first buffer on the path from $v$ to $t_i^1$ and respectively to $t_i^2$.

## 6. Experimental results

In this section we report results for a 2-pin net implementation of our algorithm; multipin nets, delay constraints, and buffer and wire sizing are currently under implementation. All experiments were conducted on an SGI Origin 2000 with 16 195MHz MIPS R10000 processors (only one of which is actually used by our sequential implementation) and 4 G-Bytes of internal memory, running under IRIX 6.5. The algorithm was coded in C and compiled using `g++` version egcs-2.91.66 with `-O4` optimization.

We tested our algorithm on the 10 circuits from [9], which were also used in [3]. Multipin nets were decomposed into 2-pin nets in the same way as in [9, 3]. The circuit parameters are summarized in Table 1.

Table 2 shows the results of the multicommodity flow algorithm (with pin assignment) when run with $D = \infty$, i.e., when the objective is to minimize the wire and buffer congestion only. The table shows that progressively better fractional solutions are obtained by the approximation algorithm. The results also show the tradeoff between congestion on one hand and wiring resources (number of buffers and wirelength) on the other hand.

Table 3 gives the results for wirelength minimization ($\alpha = 0$ and $\beta = 1$) subject to wire and buffer congestion constraints ($\mu_0 = 1.0$ and $\nu_0 = 1.0$). In these experiments the multicommodity flow algorithm is run once per testcase (without binary search), with $D$ equal to the lower bound computed by routing each net optimally without taking into account capacity constraints. The multicommodity flow runtime includes randomized rounding (10000 trials), while RABID runtime is for an RS6000/595 workstation with 1Gb of memory, as reported in [3].

The wirelength of the global routing obtained by our algorithm without pin assignment (MCF) is always within $1.03\%$ of the lower bound. In contrast, the RABID heuristic of [3] exceeds the lower bound by $5.64 - 11.87\%$. To evaluate the effect of simultaneous pin assignment, we have added the possibility for each sink to be positioned not only in the given tile, but also in the 3-8 surrounding tiles (see Table 1 for the average number of tiles per pin of each testcase). Running our algorithm with pin assignment enabled (MCF+PA) further decreases wirelength by $\approx 10\%$, while being within at most $0.15\%$ of the corresponding lower bound. We note that routing and pin assignment is performed by our algorithm in virtually the same time as routing alone.

**Table 1. Circuit parameters.**

| Circuit | #Nets | Grid size | Tile area | Avg. tiles per pin | U | #Buffer sites |
|---|---|---|---|---|---|---|
| a9c3 | 1526 | 30 x 30 | 1.09 | 4.9 | 6 | 32780 |
| ac3 | 409 | 30 x 30 | 0.49 | 5.0 | 7 | 8550 |
| ami33 | 324 | 30 x 33 | 0.46 | 5.0 | 6 | 17750 |
| ami49 | 493 | 30 x 30 | 0.68 | 4.8 | 6 | 11450 |
| apte | 141 | 33 x 30 | 0.36 | 5.0 | 7 | 4200 |
| hc7 | 1318 | 30 x 30 | 1.04 | 4.8 | 6 | 17780 |
| hp | 187 | 30 x 30 | 0.42 | 5.0 | 7 | 2350 |
| playout | 1663 | 30 x 33 | 0.78 | 4.8 | 7 | 37550 |
| xc5 | 2149 | 30 x 30 | 0.58 | 5.0 | 7 | 19150 |
| xerox | 390 | 30 x 30 | 0.38 | 5.0 | 7 | 7000 |

**Table 2. Congestion minimization results ($D = \infty$).**

| Circuit | Phase# | Wire Congest | Buffer Congest | #Buffers | Wlen | CPU sec. |
|---|---|---|---|---|---|---|
| a9c3 | 1 | 0.75 | 0.80 | 3351 | 26057 | 15.6 |
| | 4 | 0.59 | 0.43 | 3356 | 26123 | 63.0 |
| | 16 | 0.51 | 0.23 | 3402 | 26595 | 255.1 |
| | 64 | 0.46 | 0.18 | 3505 | 27328 | 1023.6 |
| | **64+ROUND** | **0.60** | **0.31** | **3606** | **27989** | **1087** |
| ac3 | 1 | 0.77 | 1.00 | 796 | 4998 | 4.0 |
| | 4 | 0.62 | 0.53 | 797 | 5008 | 15.9 |
| | 16 | 0.40 | 0.27 | 803 | 5072 | 64.4 |
| | 64 | 0.28 | 0.18 | 826 | 5211 | 260.1 |
| | **64+ROUND** | **0.46** | **0.67** | **832** | **5254** | **281** |
| ami33 | 1 | 0.66 | 0.67 | 909 | 4466 | 3.4 |
| | 4 | 0.55 | 0.36 | 908 | 4476 | 13.4 |
| | 16 | 0.47 | 0.20 | 910 | 4515 | 54.0 |
| | 64 | 0.40 | 0.14 | 930 | 4618 | 214.2 |
| | **64+ROUND** | **0.56** | **0.36** | **953** | **4703** | **234** |
| ami49 | 1 | 1.36 | 0.90 | 948 | 6045 | 4.2 |
| | 4 | 1.00 | 0.46 | 958 | 6083 | 17.1 |
| | 16 | 0.74 | 0.29 | 1040 | 6509 | 74.0 |
| | 64 | 0.66 | 0.21 | 1205 | 7278 | 304.4 |
| | **64+ROUND** | **1.00** | **0.56** | **1321** | **7767** | **329** |
| apte | 1 | 1.08 | 1.00 | 328 | 1668 | 1.6 |
| | 4 | 0.87 | 0.57 | 327 | 1677 | 6.5 |
| | 16 | 0.53 | 0.30 | 336 | 1725 | 27.0 |
| | 64 | 0.44 | 0.17 | 359 | 1836 | 112.0 |
| | **64+ROUND** | **0.69** | **1.00** | **362** | **1833** | **123** |
| hc7 | 1 | 1.00 | 1.19 | 2203 | 17670 | 10.6 |
| | 4 | 0.79 | 0.61 | 2206 | 17738 | 43.3 |
| | 16 | 0.69 | 0.31 | 2301 | 18481 | 180.1 |
| | 64 | 0.62 | 0.23 | 2498 | 19660 | 736.4 |
| | **64+ROUND** | **0.86** | **0.50** | **2678** | **20616** | **788** |
| hp | 1 | 0.92 | 1.67 | 334 | 1952 | 1.7 |
| | 4 | 0.71 | 0.85 | 330 | 1961 | 6.9 |
| | 16 | 0.46 | 0.45 | 334 | 2003 | 28.4 |
| | 64 | 0.33 | 0.29 | 355 | 2119 | 118.8 |
| | **64+ROUND** | **0.58** | **1.00** | **367** | **2153** | **131** |
| playout | 1 | 0.64 | 0.98 | 2890 | 23155 | 19.9 |
| | 4 | 0.52 | 0.42 | 2892 | 23199 | 80.5 |
| | 16 | 0.40 | 0.24 | 2922 | 23582 | 339.9 |
| | 64 | 0.33 | 0.17 | 3238 | 25809 | 1480.8 |
| | **64+ROUND** | **0.34** | **0.27** | **3453** | **27266** | **1544** |
| xc5 | 1 | 1.14 | 1.31 | 3187 | 22314 | 23.4 |
| | 4 | 0.98 | 0.66 | 3202 | 22492 | 94.3 |
| | 16 | 0.74 | 0.37 | 3277 | 23231 | 388.6 |
| | 64 | 0.66 | 0.31 | 3570 | 24872 | 1623.1 |
| | **64+ROUND** | **0.84** | **0.59** | **3929** | **26381** | **1704** |
| xerox | 1 | 0.93 | 1.42 | 659 | 3662 | 3.7 |
| | 4 | 0.72 | 0.77 | 660 | 3698 | 15.3 |
| | 16 | 0.45 | 0.40 | 684 | 3858 | 68.1 |
| | 64 | 0.32 | 0.21 | 753 | 4174 | 299.0 |
| | **64+ROUND** | **0.45** | **0.67** | **781** | **4303** | **319** |

## 7. Conclusions

In this paper we propose the first provably good approach to floorplan evaluation with simultaneous timing- and congestion-driven buffered global route planning, pin and layer assignment, and wire/buffer sizing. Preliminary experimental results show that our method significantly outperforms approaches based on cascading individual optimizations such as the recent RABID algorithm of Alpert et al. [3].

Future work aims to incorporate in our implementation practical improvements such as the use of uneven sized tiles, window constraints on buffer usage (as opposed to tile constraints), and faster-converging dual-update rules.

## 8. Acknowledgments

We thank Charles Alpert, Jason Cong, and Jiang Hu for kindly providing us with the testcases used in [9] and [3].

**Table 3. Wirelength minimization ($\alpha = 0$ and $\beta = 1$) subject to wire and buffer congestion constraints ($\mu_0 = 1.0$ and $\nu_0 = 1.0$).**

| Circuit | Algorithm | Wlen | %LB gap | #buffers | %LB gap | Wire Congest | Buffer Congest | CPU sec. |
|---|---|---|---|---|---|---|---|---|
| a9c3 | RABID | 30723 | 5.64 | 4225 | 11.95 | 0.60 | 0.44 | 502 |
| | MCF | 29082 | 0.00 | 3801 | 0.72 | 0.63 | 0.31 | 1082 |
| | MCF+PA | 26057 | 0.00 | 3376 | 0.75 | 0.58 | 0.30 | 1079 |
| ac3 | RABID | 5954 | 7.67 | 1037 | 15.74 | 0.58 | 0.33 | 208 |
| | MCF | 5530 | 0.00 | 905 | 1.00 | 0.73 | 0.67 | 280 |
| | MCF+PA | 4993 | 0.00 | 805 | 1.39 | 0.69 | 0.50 | 279 |
| ami33 | RABID | 5232 | 6.93 | 1150 | 14.20 | 0.69 | 0.44 | 138 |
| | MCF | 4893 | 0.00 | 1015 | 0.79 | 0.69 | 0.31 | 239 |
| | MCF+PA | 4464 | 0.00 | 916 | 0.88 | 0.59 | 0.25 | 237 |
| ami49 | RABID | 7592 | 11.87 | 1339 | 21.51 | 0.93 | 0.36 | 167 |
| | MCF | 6792 | 0.07 | 1135 | 2.99 | 1.00 | 0.47 | 314 |
| | MCF+PA | 6041 | 0.01 | 991 | 4.87 | 1.00 | 0.50 | 304 |
| apte | RABID | 2010 | 10.78 | 417 | 18.47 | 1.00 | 0.33 | 95 |
| | MCF | 1833 | 1.03 | 373 | 5.97 | 1.00 | 1.00 | 118 |
| | MCF+PA | 1663 | 0.15 | 330 | 4.43 | 1.00 | 1.00 | 117 |
| hc7 | RABID | 21523 | 7.54 | 2983 | 17.44 | 0.82 | 0.35 | 386 |
| | MCF | 20024 | 0.05 | 2591 | 2.01 | 0.96 | 0.45 | 775 |
| | MCF+PA | 17660 | 0.00 | 2217 | 0.82 | 0.93 | 0.54 | 767 |
| hp | RABID | 2403 | 11.12 | 450 | 20.97 | 0.83 | 0.28 | 67 |
| | MCF | 2164 | 0.06 | 395 | 6.18 | 1.00 | 1.00 | 127 |
| | MCF+PA | 1945 | 0.00 | 345 | 6.81 | 0.83 | 1.00 | 126 |
| playout | RABID | 27601 | 6.38 | 3840 | 15.04 | 0.45 | 0.64 | 813 |
| | MCF | 25946 | 0.00 | 3428 | 2.70 | 0.51 | 0.32 | 1393 |
| | MCF+PA | 23138 | 0.00 | 3004 | 4.12 | 0.40 | 0.32 | 1386 |
| xc5 | RABID | 27060 | 8.35 | 4410 | 23.25 | 0.84 | 0.81 | 694 |
| | MCF | 25155 | 0.73 | 3841 | 7.35 | 0.96 | 0.60 | 1641 |
| | MCF+PA | 22265 | 0.05 | 3340 | 4.87 | 0.98 | 0.50 | 1644 |
| xerox | RABID | 4541 | 11.48 | 957 | 30.56 | 0.93 | 0.57 | 167 |
| | MCF | 4078 | 0.12 | 807 | 10.10 | 1.00 | 0.80 | 284 |
| | MCF+PA | 3658 | 0.00 | 691 | 6.14 | 0.88 | 0.75 | 281 |

## References

[1] C. Albrecht, “Global Routing by New Approximation Algorithms for Multicommodity Flow”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, May 2001, pp. 622–632.

[2] C. Alpert and A. Devgan and S. Quay, “Is wire tapering worthwhile?”, *Proc. ICCAD*, 1999, pp.430–435.

[3] C. Alpert and J. Hu and S. Sapatnekar and P. Villarrubia, “A practical methodology for early buffer and wire resource allocation”, *Proc. DAC*, 2001.

[4] R.C. Carden and C.-K. Cheng, “A global router using an efficient approximate multicommodity multiterminal flow algorithm”, *Proc. DAC*, 1991, pp. 316–321.

[5] R. C. Carden IV and J. Li and C.-K. Cheng, “A Global Router with a Theoretical Bound on the Optimum Solution”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1996, pp. 208–216.

[6] N. P. Chen, C.-P. Hsu and E. S. Kuh, “The Berkeley building-block (BBL) layout system for VLSI design”, *VLSI83, Proceedings of the IFIP TC WG 10.5 International Conference on Very Large Scale Integration*, Trondheim, Aug. 1983, pp. 37-44.

[7] J. Cong, “Pin Assignment with Global Routing for General Cell Design,” *IEEE Trans. on CAD* 10(9) (1991), pp. 1401-1412.

[8] J. Cong, L. He, C.-K. Koh and P.H. Madden, “Performance optimization of VLSI interconnect layout”, *Integration* 21 (1996), pp. 1–94.

[9] J. Cong, T. Kong and D.Z. Pan, “Buffer block planning for interconnect-driven floorplanning”, *Proc. ICCAD*, 1999, pp. 358–363.

[10] W. W.-M. Dai and E. S. Kuh, “Simultaneous floor planning and global routing for hierarchical building-block layout”, *IEEE Trans. on CAD* 6(5) (1987), pp. 828-837.

[11] F.F. Dragan, A.B. Kahng, I.I. Măndoiu, S. Muddu and A. Zelikovsky, “Provably good global buffering using an available buffer block plan”, *Proc. ICCAD*, 2000, pp. 104–109.

[12] F.F. Dragan, A.B. Kahng, I.I. Măndoiu, S. Muddu and A. Zelikovsky, “Provably good global buffering by multiterminal multicommodity flow approximation”, *Proc. ASP-DAC*, 2001, pp. 120–125.

[13] F.F. Dragan, A.B. Kahng, I.I. Măndoiu, S. Muddu and A. Zelikovsky, “Practical approximation algorithms for separable packing linear programs”, *Proc. 7th Workshop on Algorithms and Datastructures (WADS)*, 2001, pp. 325–337.

[14] L.K. Fleischer, “Approximating fractional multicommodity flow independent of the number of commodities”, *SIAM J. Discrete Math.* 13 (2000), pp. 505–520.

[15] 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.

[16] C.A. Phillips, “The network inhibition problem”, *Proc. 25th Annual ACM Symposium on Theory of Computing*, 1993, pp. 776–785.

[17] P. Raghavan and C.D. Thomson, “Randomized rounding”, *Combinatorica*, 7 (1987), pp. 365–374.

[18] X. Tang and D.F. Wong, “Planning buffer locations by network flows”, *Proc. ISPD*, 2000.