Recent Advances in Multicommodity Flow Algorithms for Global Routing

Ion Măndoiu*
CSE Department, University of Connecticut
371 Fairfield Road, Unit 1155, Storrs, CT 06269, USA
E-mail: ion@engr.uconn.edu

Abstract

Interconnect planning and synthesis in general, and global routing in particular, are becoming critical to meeting chip performance targets in deep-submicron technologies. In addition to handling traditional objectives such as congestion, wirelength and timing, a new and very important requirement for current global routers is the integration with other interconnect optimizations, most importantly with buffer insertion and sizing.

In this paper, we review and enhance a powerful integrated approach introduced in [4] for congestion and timing-driven global routing, buffer insertion, pin assignment, and buffer/wire sizing. We extend the method to capture polarity constraints induced by inverter insertion, and present simpler and more efficient gadget constructions for buffer/wire sizing and enforcing delay constraints. Furthermore, we present experimental results detailing the scalability and limitations of proposed methods.

1 Introduction

Due to delay scaling effects in deep-submicron technologies, interconnect planning and synthesis are becoming critical to meeting chip performance targets with reduced design turnaround time [1]. In particular, the global routing phase of the design cycle is receiving renewed interest, as it must efficiently handle increasingly more complex constraints for increasingly larger designs (see [14] for a recent survey).

In addition to handling traditional objectives such as congestion, wirelength and timing, a critical requirement for current global routers is the integration with other interconnect optimizations, most importantly with buffer insertion and sizing. Indeed, it is estimated that top-level on-chip interconnect will require up to 10^6 repeaters when we reach the 50nm technology node. Since these repeaters are large and have a significant impact on global routing congestion, buffer insertion and sizing can no longer be done after global routing completes.

In this paper, we review and enhance a powerful integrated approach introduced in [4] for congestion and timing-driven global routing, buffer insertion, pin assignment, and buffer/wire sizing. Our approach is based on a multicommodity flow formulation for the buffered global routing problem. Multicommodity flow based global routing has been an active research area since the seminal work of Raghavan and Thomson [16]. Although the global routing problem is NP-hard (even highly restricted versions of it, see [19]), [16] has shown that the optimum solution can be approximated arbitrarily close in time polynomial in the number of nets and the inverse of the accuracy. To date, predictability of solution quality continues to be a distinct advantage of multicommodity flow based methods over all other approaches to global routing, including popular rip-up-and-reroute approaches [14].

The original method of Raghavan and Thomson relies on randomized rounding of an *optimum* fractional multicommodity flow. Subsequent works [8, 15] have improved runtime scalability by using the approximation algorithm for multicommodity flows by [17]. Yet, only the recent breakthrough improvements due to Garg and Könemann [13] and Fleischer [12] have rendered multicommodity flow based global routing practical for full chip designs [3]. As [3], our algorithm is built upon the efficient multicommodity flow approximation scheme of [13, 12].

In next section we review the multicommodity flow approach as applied to buffered global routing in [4], highlighting the gadget graph construction used to capture valid buffered routes in the context of a (set capacitated) multicommodity flow formulation. The main contribution of the paper is represented by simpler and more efficient gadget constructions for capturing polarity constraints induced by inverter insertion, buffer and wire sizing, and delay constraints (Section 4). Unlike the constructions in [4], none of the simplified constructions requires changes to the algorithm for multicommodity flow approximation. We conclude the paper with experimental results detailing the scalability and limitations of the proposed methods and with directions for future research.

2 Problem Formulation

In this section we give an integrated formulation for the global routing and the bounded wireload buffer insertion problems. Polarity constraints induced by the use of inverting buffers, buffer and wire sizing, and timing constraints are individually discussed in Section 4.

As in [3], we capture wire and buffer congestion using a *tile graph* $G = (V, E)$ which has an edge between any two adjacent tiles (see Figure 1), together with buffer and wire capacity functions $b : V \rightarrow \mathbf{N}$ and $w : E \rightarrow \mathbf{N}$, respectively. For each tile $v \in V$, the *buffer capacity* $b(v)$ is the number of buffer sites located in v. Similarly, 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.

*Work performed while the author was with the Electrical and Computer Engineering Department at the University of California, San Diego.

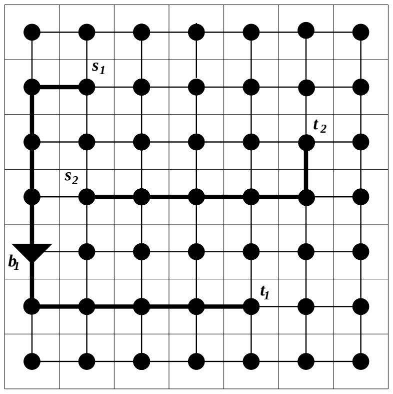

Figure 1: Tile graph with two 2-pin nets.

Let $N_1, N_2, \ldots, N_k$ be the given nets, where each net N_i is specified by a *source* s_i and a *sink* t_i. For each net N_i, we seek an s_i–t_i path P_i in G buffered using only available buffer sites 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:

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

Given feasible buffered routings (P_i, B_i) for each net N_i, the relative buffer and wire congestion are defined by

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

and

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

respectively. The buffered paths (P_i, B_i), $i = 1, \ldots, k$, are simultaneously routable if and only if both $\mu \leq 1$ and $\nu \leq 1$. Using a linear combination of total wirelength and buffer count as solution quality measure, we get the following formulation:

Integrated Global Routing and Bounded Wireload Buffer Insertion Problem[1]
Given:

- Grid-graph $G = (V, E)$, with buffer and wire capacities $b : V \rightarrow \mathbf{N}$, respectively $w : E \rightarrow \mathbf{N}$;
- 2-pin nets $N_1, \ldots, N_k$, each net N_i with a source pin s_i and a sink pin t_i; and
- Wireload upper-bound $U > 0$

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

[1]The problem is called *Floorplan Evaluation Problem* in [4], but the formulation is useful in post-placement scenarios as well.

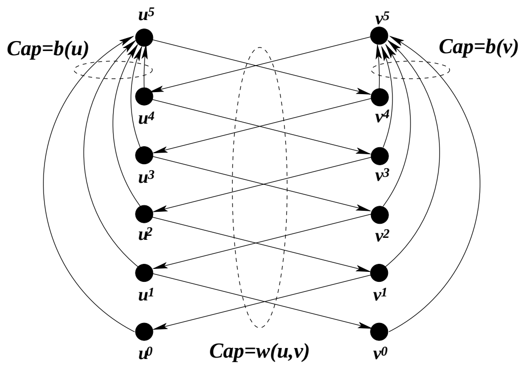

Figure 2: The basic gadget replacing edge (u, v) of the tile graph for buffer wireload upperbound $U = 5$.

3 Solution Based on Multicommodity Flow Approximation

The high-level steps in the multicommodity flow based approach in [4] are the following:

1. Build an auxiliary graph in which every directed path from a net source to the net's sink captures a feasible wire route between them together with locations for the buffers to be inserted on this route such that buffer load constraints are satisfied. The auxiliary graph is obtained automatically from the tile graph using a *gadget construction*.
2. Use the auxiliary graph to formulate the floorplan evaluation problem as an integer linear program (ILP). To formally express the ILP, we use a 0/1 variable for each source-sink path, and require that exactly one path be chosen for each source-sink pair. The objective is to minimize the wire and buffer congestion subject to a given upper-bound on the total wirelength.
3. Find a near-optimal solution to the fractional relaxation of the above integer program using the general framework for multicommodity flow approximation of [13, 12]. Although the integer program has exponential size (there are exponentially many variables corresponding to source-sink paths in the auxiliary graph), the algorithm still runs in polynomial time by representing explicitly only non-zero variables.
4. Finally, use randomized rounding [16] to convert the fractional multicommodity flow to an integer one.

To facilitate understanding of the constructions given in Section 4, we discuss here in detail the construction of the auxiliary directed graph H which captures feasible buffered routings (see Figure 2). 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. The graph H has $U + 1$ vertices $v^0, v^1, \ldots, v^U$ for each vertex $v \in V(G)$. The index of each copy corresponds to the *remaining wireload budget*, i.e., the number of units of wire that can still be driven by the last inserted buffer (or by the net's source). Buffer insertions are represented in the gadget graph by directed arcs of the form (v^j, v^U): following such an arc resets the remaining wireload budget up to the maximum value of U. Each undirected edge (u, v) in the tile graph gives rise to directed arcs (u^j, v^{j-1}) and (v^j, u^{j-1}),

$j = 1, \ldots, U$, in the gadget graph. Notice that the copy number decreases by 1 for each of these arcs, corresponding to a decrease of 1 unit in the remaining wireload budget. In addition, we add to H individual vertices to represent net sources and sinks. Each source vertex is connected by a directed arc into the U-th copy of the node representing the enclosing tile. Furthermore, *all* copies of the nodes representing enclosing tiles are connected by directed arcs into the respective sink vertices.

Formally, the graph H has vertex set

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

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

$$\begin{array}{lcl} E_{src} & = & \{(s_i, v^U) \mid \text{tile } v \text{ contains } s_i, 1 \le i \le k\} \\ E_{sink} & = & \{(v^j, t_i) \mid \text{tile } v \text{ contains } t_i, 0 \le j \le U, 1 \le i \le k\} \\ E_{u,v} & = & \{(u^j, v^{j-1}), (v^j, u^{j-1}) \mid 1 \le j \le U\} \\ E_v & = & \{(v^j, v^U) \mid 0 \le j < U\} \end{array}$$

Each directed path in the gadget graph H corresponds to a buffered routing in the tile graph, obtained by ignoring copy indices for tile vertices and replacing each"buffer" arc (v^j, v^U) with a buffer inserted in tile v. Clearly, the construction ensures that the wireload of each buffer is at most U since a directed path in H can visit at most U vertices before following a buffer arc.

Let $\mathcal{P}_i$ denote the set of all simple s_i–t_i paths in H. To get an ILP formulation, 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, the integrated global routing and bounded wireload buffer insertion problem can be formulated as follows:

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

subject to

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

ILP (1) is similar to the "path" formulation of the classical minimum cost integer multicommodity flow problem [2]. The only difference is that capacity constraints on the edges and vertices of the tile graph G become capacity constraints for sets of edges of the gadget graph H (see Figure 2). We note that the floorplan evaluation problem can be represented more compactly by using a polynomial number of edge-flow variables instead of the exponential number of path-flow variables x_p. However, we use formulation (1) since it leads to stronger fractional relaxations [9]. The exponential number of variables is not impeding the efficiency of the approximation algorithm, which, during its execution, represents explicitly only a polynomial number of paths with non-zero flow.

Instead of solving the relaxation of ILP (1) directly [4] introduces an upper bound D on the wire and buffer area and considers the following related linear program (LP):

$$\min \lambda \qquad (2)$$

subject to

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

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

Let λ^* 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 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., $D\text{max} = \alpha \sum_{v\in V(G)} b(v) + \beta \sum_{(u,v)\in E(G)} w(u,v)$. In particular, unfeasibility of the fractional relaxation of ILP (1) is equivalent to λ^* being greater than 1 when $D = D\text{max}$, and can therefore be detected by an approximation algorithm for (2).

In the interest of space we omit the details of the algorithm for approximating the optimum solution to LP (2), and direct the interested reader to [4].

4 Extensions

In this section we show how to extend the multicommodity flow approach to handle polarity constraints imposed by the use of inverting buffers, buffer and wire sizing, and prescribed delay upperbounds (for the extension to pin assignment see [4, 5]). Polarity constraints have not been considered in [4], while the constructions presented here for buffer/wire sizing and for enforcing delay upperbounds are simpler and more efficient than the original constructions in [4]. In particular, unlike the constructions in [4], the constructions given here involve only changes to the gadget graph, leaving the approximation algorithm used to solve LP (2) unchanged.

4.1 Polarity Constraints

The basic problem formulation in Section 2 considers only a non-inverting buffer type. In practice, inverting buffers are often preferred since they occupy a smaller area for the same driving strength. Although the use of inverting buffers introduces additional *polarity constraints*, which may require a larger number of buffers to be inserted, inverting buffers are likely to lead to better overall resource utilization. Algorithms for bounded capacitive load inverting (and non-inverting) buffer insertion have been recently discussed in [7]. The focus of [7] is on single net

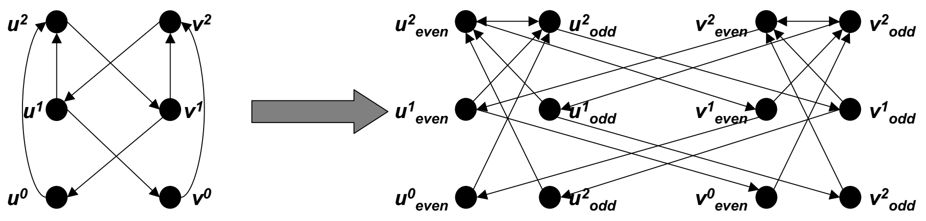

Figure 3: Gadget for polarity constraints with buffer load upperbound $U = 2$.

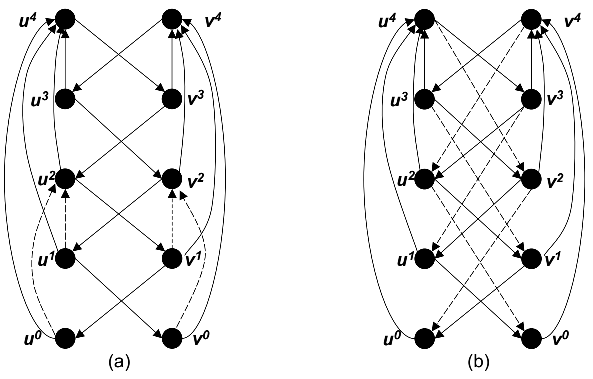

Figure 4: (a) Gadget for buffer sizing with two available buffer sizes, one with wireload upperbound $U = 4$ and one with wireload upperbound $U = 2$. Solid arcs (u^i, u^4), respectively (v^i, v^4), correspond to the insertion of a buffer capable of driving 4 units of wire, while dashed arcs (u^i, u^2) and (v^i, v^2) correspond to the insertion of a smaller buffer capable of driving 2 units of wire. (b) Gadget for wire sizing with two available wire widths, standard width and "half" width (i.e., wire with double per unit capacitive load). Solid arcs (u^i, v^{i-1}) and (v^i, u^{i-1}) correspond to standard width connections between tiles u and v, while dashed arcs (u^i, v^{i-2}) and (v^i, u^{i-2}) correspond to half-width connections.

buffering, with arbitrary positions for the buffers. Here, our goal is to minimize the overall number of buffers required by the nets, and to ensure that buffers are inserted only in the available sites.

Consideration of polarity constraints is achieved by modifying the basic gadget graph given in Section 2 as follows (see Figure 3). Each node of the basic gadget is replaced by an "even" and "odd" copy, i.e., v^i is propagated into v^i_{even} and v^i_{odd}. Tile-to-tile arcs are replaced by two arcs connecting copies with the same polarity, e.g., the arc (u^i, v^{i-1}) gives rise to $(u^i_{even}, v^{i-1}_{even})$ and $(u^i_{odd}, v^{i-1}_{odd})$. If a path uses such an arc, then it does not change polarity. Instead, each buffer arc changes polarity, i.e., (v^i, v^U) gives rise to (v^i_{even}, v^U_{odd}) and (v^i_{odd}, v^U_{even}). The gadget also allows two inverting buffers to be inserted in the same tile for the purpose of meeting polarity constraints. This is achieved by providing bidirectional arcs connecting the U-th even and odd copies of a tile v, i.e., (u^U_{even}, u^U_{odd}) and (v^U_{odd}, v^U_{even}). Finally, source vertices s_i are connected by directed arcs into the even U-th copy of enclosing tiles, and only copies of the desired polarity are connected by arcs to sink vertices t_i.

4.2 Buffer and Wire Sizing

Buffer and wire sizing are well-known techniques for timing optimization in the final stages of the design cycle [10]. However, buffer and wire sizing can be equally effective for reducing congestion and/or wiring resources. In this section we show how to incorporate buffer and wire sizing in the multicommodity flow framework. The key enablers to these extensions are again appropriate modifications of the gadget graph.

The gadget for buffer sizing is illustrated in Figure 4(a) for two available buffer sizes, one with wireload upperbound $U = 4$ and one with wireload upperbound $U = 2$. The general construction entails using a number of copies of each tile vertex equal to the maximum buffer load upperbound U. For every buffer with wireload upperbound of $U' \leq U$, we insert buffer arcs $(v^i, v^{U'})$ for every $0 \leq i < U'$. Thus, the copy number of each vertex continues to capture the remaining wireload budget, which ensures the correctness of the construction.

Wire sizing can be handled by a different modification of the gadget graph (see Figure 4(b)). Assuming that per unit capacitances of the thinner wire widths are rounded to integer multiples of the "standard" per unit capacitance, the gadget models the use of thinner segments of wire by providing tile-to-tile arcs which decrease the tile copy index (i.e., remaining wireload budget) by more than one unit. For example, in Figure 4(b), solid arcs (u^i, v^{i-1}) and (v^i, u^{i-1}) correspond to standard width connections between tiles u and v, while dashed arcs (u^i, v^{i-2}) and (v^i, u^{i-2}) correspond to "half-width" connections, i.e., connections using wire with double capacitive load per unit.

4.3 Delay Constraints

[4] proposed a method for enforcing given sink delay constraints based on charging wiresegment delays to buffer arcs in the gadget graph, and using a routine for computing minimum-weight *delay constrained paths* in the algorithm for approximating the fractional solution to ILP (1). Here we give a different method for handling sink delay constraints. The new method is similar in spirit to the constructions in previous sections, relying exclusively on a modification of the gadget graph.

In general, our construction applies for any delay model, such as the Elmore delay model, for which (1) the delay of a buffered path is the sum of the delays of the path segments separated by the buffers, and (2) the delay of each segment depends only on segment length and buffer parameters. However, for the sake of efficiency, segment delays would have to be rounded to relatively coarse units.

Figure 5 shows the gadget construction for the case when delay is measured simply as the number of inserted buffers. The idea is again to replicate the basic gadget construction, this time a number of times equal to the maximum allowed net delay. Within each replica, tile-to-tile arcs decrease remaining wireload budget by one unit. In order to keep track of path delays, buffer arcs advance over a number of gadget replicas equal to the delay of the wiresegment ended by the respective buffer (this delay can be easily determined for each buffer arc since the tail of the

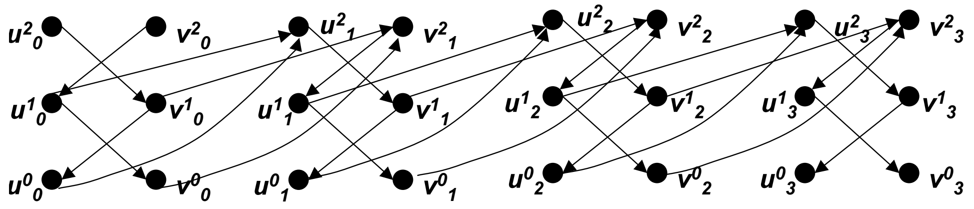

Figure 5: Gadget for enforcing delay constraints when the delay is measured by the number of buffers inserted between source and sink. The basic gadget is replicated a number of times equal to the maximum allowed net delay (3 in this example). Tile-to-tile arcs decrease remaining wireload budget within a gadget replica, while buffer arcs advance from one replica to the next.

Table 1: Circuit parameters.

Circuit	# 2-Pin Nets	Grid size	Tile area	$w(e)$	Avg. tiles per pin	U	#Buffer sites
a9c3	1526	30 x 30	1.09	52	4.9	6	32780
ac3	409	30 x 30	0.49	26	5.0	7	8550
ami33	324	33 x 30	0.46	32	5.0	6	17750
ami49	493	30 x 30	0.68	14	4.8	6	11450
apte	141	30 x 33	0.36	13	5.0	7	4200
hc7	1318	30 x 30	1.04	28	4.8	6	17780
hp	187	30 x 30	0.42	12	5.0	7	2350
playout	1663	33 x 30	0.78	120	4.8	7	37550
xc5	2149	30 x 30	0.58	50	5.0	7	19150
xerox	390	30 x 30	0.38	40	5.0	7	7000

Table 2: Wirelength minimization results for non-inverting vs. inverting buffer insertion. The number of buffer sites was assumed to be the same in both experiments.

Testcase	Non-inverting buffers			Inverting buffers		
	Wlen	#buffers	CPU	Wlen	#buffers	CPU
a9c3	29082	3800	775	29082	4540	1470
ac3	5530	905	204	5530	1095	417
ami33	4893	1014	177	4893	1186	359
ami49	6792	1133	227	6790	1417	449
apte	1833	377	88	1833	441	185
hc7	20024	2591	551	20024	3358	1030
hp	2165	404	95	2164	495	201
playout	25946	3429	1002	25946	4235	1982
xc5	25151	3843	1162	25222	4799	2285
xerox	4078	805	212	4155	1050	520

arc fully identifies the length of the wiresegment). The construction is completed by connecting net sources to the vertices with maximum remaining wireload budget in the "0 delay" replica of the gadget graph, and adding arcs into the sinks from all vertices in replicas corresponding to delays smaller than the given delay upperbounds.

5 Experimental Results

In this section we report results for a C implementation of our 2-pin net multicommodity flow based algorithm. All experiments have been conducted on a 360 MHz SUN Ultra 60 workstation with 2 GB of memory, running under SunOS 5.7. We present results for the 10 circuits in [6, 4], which are derived from testcases first used by [10]. For a more comprehensive set of results – in particular, for results on integrated pin assignment and a comparison of our method to the RABID algorithm of [6] – we direct the reader to [5].

Circuit parameters are summarized in Table 1. As in [6] and [10], we decomposed multipin nets into 2-pin nets by making direct connections from the source of a net to each of the net's sinks.

Tables 2 gives results for the extension of the multicommodity flow algorithm to inverting buffer insertion, which is about twice slower than non-inverting buffer insertion due to the doubling in size of the gadget graph. Inverter insertion leads to a very small increase in the number of buffers (due to the need to satisfy polarity constraints) which is easily compensated by the smaller size of inverters. At the same time, inverter insertion requires virtually the same wirelength (and often gives improved congestion, see [5]).

Table 3 gives runtime scaling results for the extension of the multicommodity flow algorithm to delay constraints. We note that the algorithm becomes faster for very tight delay constraints, since the number of nets that can meet delay constraints is only a fraction of the total number of nets. For moderately tight delay constraints almost all nets become routable, yet the runtime is comparable to that of the unconstrained version of the algorithm. For very lax delay constraints all nets become routable, and the runtime becomes significantly higher than that of the delay-oblivious version of the algorithm, by a factor roughly proportional to the increase in the size of the gadget graph, i.e., the maximum delay upperbound. However, large delay constraints are not very useful since they are satisfied almost in totality by using the unconstrained version of the algorithm.

To further explore the scalability of our multicommodity flow algorithm, in Table 4 we report its memory usage and runtime as a function of the grid size and of the precision parameter ε. We note that memory usage grows roughly proportional to the grid size, while the runtime does not always follow the quadratic dependence on $1/\varepsilon$ since, as in all previous experiments, the number of iterations was limited to no more than 64.

6 Conclusions

In this paper we have reviewed the powerful multicommodity flow based approach to integrated congestion and timing-driven global routing, buffer insertion, pin assignment, and buffer/wire sizing introduced in [4]. We have extended the method to capture polarity constraints induced by inverter insertion, and have presented simpler and more efficient gadget constructions for buffer/wire sizing and enforcing delay constraints.

An important direction for future research is to find practical extensions of the multicommodity flow approach to multipin

Table 3: Runtime scaling for the timing-driven version of the MCF algorithm (delay measured by number of inserted buffers).

Testcase	Delay bound = 1		Delay bound = 2		Delay bound = 4		Delay bound = 8		No delay bound	
	#nets	CPU	#nets	CPU	#nets	CPU	#nets	CPU	#nets	CPU
a9c3	455	77	820	361	1361	2178	1526	7667	1526	775
ac3	152	29	249	122	374	666	409	2270	409	204
ami33	63	13	125	50	260	413	323	1761	324	177
ami49	177	25	298	113	442	598	493	2161	493	227
apte	49	12	67	33	126	255	141	968	141	88
hc7	569	70	873	305	1231	1584	1318	5217	1318	551
hp	76	15	124	63	174	330	187	1083	187	95
playout	657	124	1095	651	1575	3506	1663	10979	1663	1002
xc5	1072	192	1429	748	2100	4158	2149	12351	2149	1162
xerox	163	29	282	201	360	752	390	2308	390	212

nets. Unfortunately, the solution proposed in [4] for 3-pin nets becomes impractical when generalized to larger net sizes.

7 Acknowledgments

I would like to thank C. Albrecht, A.B. Kahng, and A. Zelikovsky, with whom most of the presented ideas have been developed. I would also like to thank C. Alpert, J. Cong, and J. Hu for providing the testcases used in experiments.

Table 4: Memory usage (Mb) and runtime (CPU seconds) for the MCF algorithm on testcase a9c3 as a function of ε and grid size. The number of iterations was bounded to 64.

ε	10×10		20×20		30×30		40×40		50×50	
	Mb	CPU	Mb	CPU	Mb	CPU	Mb	CPU	Mb	CPU
0.9	14	32	51	58	111	195	195	664	304	2094
0.6	14	31	51	93	111	396	195	1598	304	4226
0.3	14	45	51	297	111	775	195	1925	304	4283

References

[1] International Technology Roadmap for Semiconductors, http://public.itrs.net/ (2002 Update).

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

[3] C. Albrecht, “Global Routing by New Approximation Algorithms for Multicommodity Flow”, *IEEE Trans. on CAD* 20 (2001), pp. 622–632.

[4] C. Albrecht, A. B. Kahng, I. I. Măndoiu and A. Zelikovsky, “Floorplan Evaluation with Timing-Driven Global Wireplanning, Pin Assignment, and Buffer/Wire Sizing”, *Proc. Intl. Conf. on VLSI Design/ASP-DAC*, Jan. 2002, pp. 580-587.

[5] C. Albrecht, A. B. Kahng, I. I. Măndoiu and A. Zelikovsky, “Floorplan Evaluation with Timing-Driven Global Wireplanning, Pin Assignment, and Buffer/Wire Sizing”, submitted to *IEEE Trans. on CAD*.

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

[7] C. Alpert, A.B. Kahng, B. Liu, I.I. Măndoiu, and A.Z. Zelikovsky. “Minimum buffered routing with bounded capacitive load for slew rate and reliability control”, *IEEE Trans. on CAD* 22 (2003), pp. 241–253.

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

[9] S. Chopra, “Comparisons of formulations and a heuristic for packing Steiner trees in a graph”, *Annals of Oper. Res.* 50 (1994), pp. 143–171.

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

[11] F. F. Dragan, A. B. Kahng, I. I. Măndoiu, S. Muddu and A. Zelikovsky, “Provably Good Global Buffering by Generalized Multiterminal Multicommodity Flow Approximation”, *IEEE Trans. on CAD* 21 (2002), pp. 263–274.

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

[13] N. Garg and J. Könemann, “Faster and simpler algorithms for multicommodity flow and other fractional packing problems”, *Proc. 39th Annual Symp. on Foundations of Computer Science*, 1998, pp. 300–309.

[14] J. Hu and S. Sapatnekar, “A survey on multi-net global routing for integrated circuits”, *Integration* 31 (2001), pp. 1–49.

[15] 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.

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

[17] F. Shahrokhi and D. W. Matula, “The Maximum Concurrent Flow Problem”, *J. Assoc. Comput. Mach.* 37(2), Apr. 1990, pp. 318–334.

[18] X. Tang and D.F. Wong, “Planning buffer locations by network flows”, *Proc. ISPD*, 2000, pp. 180–185.

[19] J. Vygen, *Theory of VLSI Layout*, Habilitation thesis, University of Bonn, 2001.