Net Reordering and Multicommodity Flow Based Global Routing for FPGAs

Cristinel Ababei
Department of Electrical and Computer Engineering
Marquette University, Milwaukee WI, USA
Email: cristinel.ababei@marquette.edu

Rajesh G. Kavasseri, Mohammad A. Zare
Department of Electrical and Computer Engineering
North Dakota State University, Fargo ND, USA
Email: {rajesh.kavasseri,m.zare}@ndsu.edu

***Abstract*—The most popular algorithm for solving the routing problem for field programmable gate arrays (FPGAs) has virtually remained the same for the past two decades. It is essentially an iterative maze technique, such as Dijkstra's algorithm, applied to each net in the circuit repeatedly. During multiple routing iterations, nets are ripped-up and rerouted via different paths to resolve competition for routing resources or to improve circuit delay. The most popular implementation of such a routing approach is the PathFinder algorithm used inside the VPR tool [1]. The quality of the routing solution depends however on the order in which nets are processed during each of the routing iterations. This is commonly referred to as the net ordering problem. PathFinder addresses this problem through continuous updates of the cost associated with overusing routing resources. After each routing iteration, the cost of overusing a routing resource is increased based on the routing so far, so that probability of resolving all congestion during future iterations increases.**

To further address the net ordering problem, in this paper, we investigate the effectiveness of two combined techniques to enhance PathFinder. We change the order in which nets are ripped-up and rerouted to give higher priority to nets with two, three, and more than eleven pins because these nets have the largest impact on the quality of the routing solution. Also, we alter the cost calculation during wave expansions for two-pin nets based on the global routing solution obtained by solving an equivalent multicommodity flow problem. Preliminary results suggest that the conventional FPGA routing solutions can still be improved.

***Keywords*-FPGAs; global routing; net reordering; multicommodity flow;**

I. Introduction

Routing is an essential step in the design automation of standard-cell based very large scale integration (VLSI) and FPGA circuits. It is a back-end design step where routing paths must be found for each of the circuit nets by utilizing a limited number of routing resources. Typically, this is accomplished by routing each net sequentially. The main issue with this approach has always been that the quality of the solution depends on the order in which nets are processed, and that it is hard to find a good net ordering [3]. The most popular approach to address the net ordering issue is the rip-up and reroute method. In this method, each net is routed first individually without considering congestion, usually constructing Steiner minimum trees for each net. After all of the nets have been routed, the congested areas are identified and the nets in those areas are ripped-up and rerouted through less congested areas. In the case of VLSI circuits, this approach is used during both global and detailed routing steps. Global routing, which is done first on a coarser routing graph, serves as a guide and helps to speed up and improve the quality of the detailed routing step. Because rip-up and reroute based global routing also suffers from the net ordering issue, multicommodity flow algorithms have been studied to address this problem [4].

978-1-4799-5944-0/14/$31.00

In the case of FPGA circuits, the most successful routing approach has also been the rip-up and reroute method. For instance, the widely popular academic place and route tool, VPR [1], [2], uses an improved version of the Pathfinder routing algorithm [5]. However, despite the ever growing size and complexity of FPGA circuits, routing has remained largely a one-level rip-up and reroute approach, with increased computational runtimes and unknown quality gap between the actual and the optimal routing solutions. In this paper, we revisit routing for FPGAs. Specifically, we investigate whether multicommodity flow (MCF) based global routing can also be used in the context of FPGAs to guide the detailed routing step in an effort to improve the quality of the final routing solution. To further mitigate the net ordering problem, we change the order in which nets are ripped-up and rerouted to give give higher priority to nets with two, three, and more than eleven pins because these nets have the largest impact on the routing solution.

II. Previous Work

As the most successful routing algorithm for FPGA circuits, PathFinder [5] introduced the idea of negotiated congestion. It repeatedly rips-up and reroutes each net in the circuit until all congestion is resolved. In this way it converges to a solution in which all nets are routed and no routing resource is overused. Routability is achieved by forcing nets to negotiate for a routing resource and thereby determine which net needs the resource the most. Delay is minimized by allowing the more critical nets a greater say in this negotiation. The VPR router improves the performance of Pathfinder through smarter maze routing wavefront expansions [1], [2].

These routers are essentially *detailed* routing algorithms. Some authors however do utilize the term *global* routing to refer to the portion of the above algorithms that is responsible

with the rip-up decisions and the calls of the core routine that performs wavefront expansions for individual nets [1]. While some authors distinguish between global and detailed routing [6], others refer to the entire routing problem for FPGAs as global routing [7].

Even though global routing was studied for FPGAs before [6], it is not clear what is the impact of a global routing step on the quality of the final detailed routing solution. For instance, Haritaoglu and Ayakanat [6] report results in terms of channel densities for testcases that are not part of the widely popular VPR tool. To address the net ordering problem, Lee et al. [8] propose to route simultaneously all nets connected to a given logic block using a min-cost flow computation based routing method. To alleviate the block ordering problem, they use an iterative refinement scheme based on Lagrangian relaxation. However, results are reported only for nine small testcases which are different from the twenty testcases included in the VPR tool package [10]. The study in [9] introduces a genetic algorithm based routing approach which selects the best routing solution for each net from a set of pre-computed routings. They report results that are similar to previous routing approaches. While the VPR placement algorithm has been improved in several occasions (see for instance [11]), the VPR routing remained the same during almost the last two decades.

III. Relation Between Structure of Testcases and the Quality of Routing

The quality of routing[1] is affected by multiple factors. To gain insights into how the distribution of the number of pins of each net as well as the number of nets affect the circuit delay, we have modified the VPR tool and used it to perform an investigation whose results are reported and discussed here. This investigation is done using the testcases listed in Table I from Section V-B. These testcases include all twenty testcases of VPR 4.3 [10]. All simulations are run using default values for the user-controlled parameters of the VPR tool.

The analysis data are reported in the plots from Fig.1, where the values reported in Fig.1.b and Fig.1.c represent averages of all individual values obtained for each of the testcases from Table I. It can be observed that the circuit netlists are formed mostly of nets with a small number of pins (i.e., two and three pins) as seen in Fig.1.a. The number of pins includes the source and all sinks of a given net. Consequently, the aggregated contribution of all these nets' delay to the total circuit delay is the most significant as shown in Fig.1.b. However, large nets with more than eleven pins contribute noticeably as well to the total circuit delay as seen in Fig.1.b. This contribution becomes even more significant at the expense of a lesser contribution from nets with two and three pins when we look at only the critical-path in the circuit (see Fig.1.c).

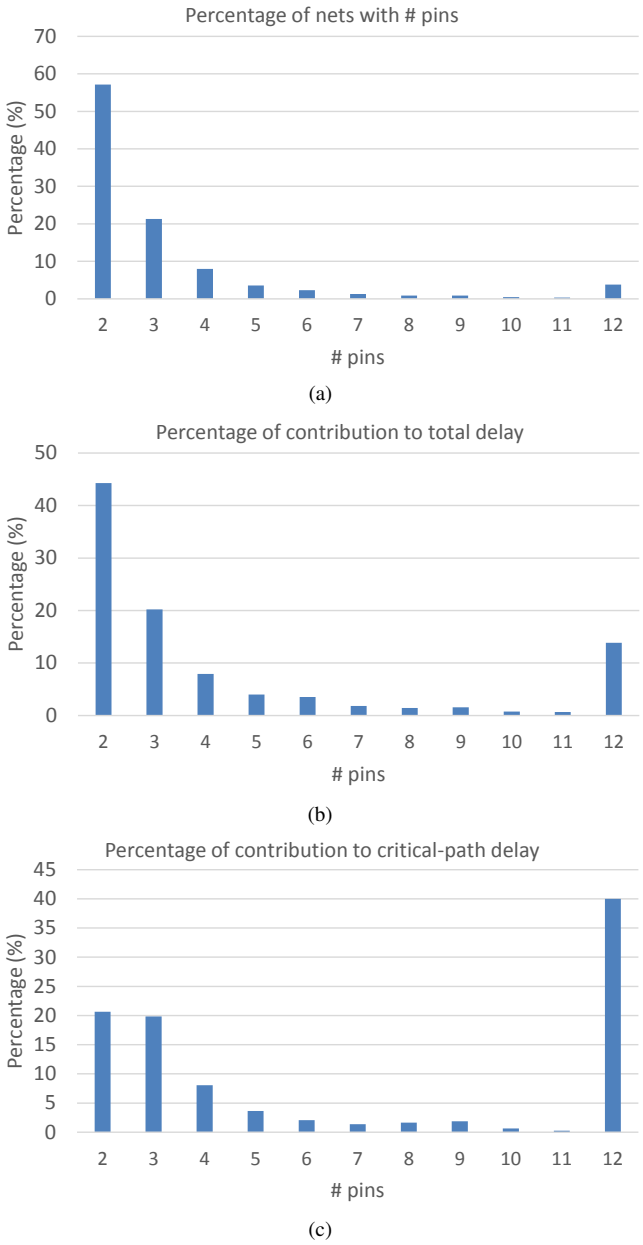

Fig. 1. (a) Percentage of nets whose number of pins is $\{2, 3, ..., 11, \geq 12\}$, (b) Contribution of nets' delay to the total circuit delay, (c) Contribution of nets' delay to the critical-path delay in the circuit.

These plots suggest that one needs to concentrate primarily on the nets with two and three pins and nets with more than eleven pins in order to have a first order impact on the total and critical-path delays. These insights motivate us to investigate the following techniques to enhance the quality of routing:

1) We change the conventional random net ordering during detailed routing to give higher priority to nets with two and three pins as well as to nets with more than eleven pins. One can see this technique as a form of *preconditioning of net ordering*.

[1]Quality of routing can be measured as the final circuit delay after detailed routing or as the minimum channel-width required for successful routing or the computational runtime of the routing algorithm (directly affected by the total number of iterations until successful routing is achieved) or total wirelength (WL).

2) Because two-pin nets represent the vast majority of nets, we introduce a multicommodity flow based *global routing* step which processes only these nets. An efficient approximation algorithm is used to solve the equivalent multicommodity flow problem, thereby *simultaneously* and globally routing these nets. The global routing result is used later during the detailed routing step to modify the internal expansion costs such that detailed routing paths follow as closely as possible the global routes. This technique mitigates the net ordering issue for two-pin nets.

IV. Global Routing as a Multicommodity Flow Problem

The proposed global routing for two-pin nets is similar to the conventional global routing used in standard-cell based digital design flows. It is formulated on a global routing graph (GR-graph), which in this case is constructed as illustrated in Fig.2. The entire FPGA area is partitioned into a number of tiles as indicated by the hashed lines in Fig.2.b. Each tile has an associated node in the GR-graph. Arcs of the GR-graph connect only nodes corresponding to adjacent tiles. Thus, the GR-graph is a regular mesh graph. The tiles on the four edges of the FPGA area include also the IO pads as shown in Fig.2.b. Each arc is assigned a capacity which depends on the *channel-width* internal variable of the VPR tool. The channel-width represents the number of routing resources in a vertical or horizontal routing channel.

The global routing problem is defined as the problem of finding routing paths on the GR-graph for all two-pin nets from the netlist of a given testcase such that arcs' capacities are not violated. This problem is solved by mapping it into a multicommodity flow (MCF) problem, which is then solved using an efficient polynomial time approximation algorithm.

Multicommodity flow problems represent a class of network flow problems in which it is necessary to distinguish among the flows in the network [12]. More formally, the MCF problem is formulated on a given network of nodes connected by edges. Each edge has a particular capacity (a.k.a. "bandwidth") u, as well as a cost c associated with it. Given is also a set K of k commodities, where a commodity i is defined by a triple (s_i, t_i, d_i) – source, sink and demand. Traffic can flow along each edge consuming some of the bandwidth. The total traffic flow is made up of all node-to-node demands that can go through multiple hops along their routes in order to get from source to destination. The objective of the MCF problem is to find a feasible and optimal (i.e., minimum cost) set of routes through the network for each of those demands, subject to joint capacity constraints. In addition, the traffic demands cannot be split onto different routes, that is, the complete connection for any commodity must be routed through a single path.

In our approach, the GR-graph is utilized as the network to formulate the MCF problem. This is achieved by associating a new commodity in the MCF problem to each two-pin net from the global routing problem. For instance, the MCF problem with three commodities from Fig.3 is constructed for a global routing problem which must route three two-pin nets.

The MCF problem is solved using an efficient polynomial time approximation algorithm studied in [13], [14]. This polynomial time approximation algorithm is based on the recent advancements in polynomial time approximation schemes (PTAS) [15] and can achieve $(1+\epsilon)$ optimal solutions, where ϵ is an input accuracy threshold. The basic idea of this algorithm is that it proceeds in phases and each phase is composed of a number of k iterations. In iteration j of the i^{th} phase we route d_j units of commodity j in a sequence of steps. In each step, a shortest path from source s_j to destination t_j is computed using a so called length function. The solution found using this algorithm is a *fractional* flow solution, meaning that the flow for a given source-destination pair may be split between multiple paths. Because we cannot use directly such a fractional flow solution, we use a simple heuristic to round the fractional flows to *integral* flows such that only one routing path is selected for each source-destination pair. The heuristic simply rounds flows along the paths with highest fractions. Since the polynomial time approximation algorithm itself is not part of the actual contribution of this paper, we refer the reader to [13], [14] for further implementation details.

The solution of the MCF problem found by the above polynomial time approximation algorithm represents essentially the solution to the global routing problem. This solution can be utilized to guide the detailed routing step later on such that detailed routing paths follow as closely as possible the global routing solution. For instance, the detailed routing path for the two-pin net whose source is $CLB(1,1)$ and sink is $CLB(3,3)$ shown in Fig.2.b follows the global routing solution given by the sequence of nodes $\{1,4,5,6,9\}$ which is found as the path though which the associated commodity is transported in the MCF problem formulation.

V. Enhanced PathFinder

A. Enhanced PathFinder Routing Algorithm

The proposed modified PathFinder routing algorithm is shown in Fig.4. In the first phase, the MCF problem is constructed for two-pin nets. Commodities in the MCF problem formulation are associated to two-pin nets as discussed in the previous section. Then, the MCF problem instance is solved using the polynomial time approximation algorithm. The global routing paths for all commodities represent *preferred* routing paths for the corresponding two-pin nets. PathFinder is changed in the second phase to guide the routing of the two-pin nets to follow as closely as possible the preferred global routing paths. The change consists in penalizing more the usage of routing resources which are outside the preferred global routing paths during wavefront expansions of the detailed routing. The order in which nets are ripped-up and rerouted is also changed. Nets with 2, 3, and greater than 11 pins are processed after the rest of the nets during each outer loop iteration. In this way, nets with a number of pins between 4 and 11 are always ripped-up and rerouted first during a new iteration while nets with 2, 3, and greater than 11 pins are left with the routing solution from a previous iteration if their routing was successful. Because initially all nets are routed

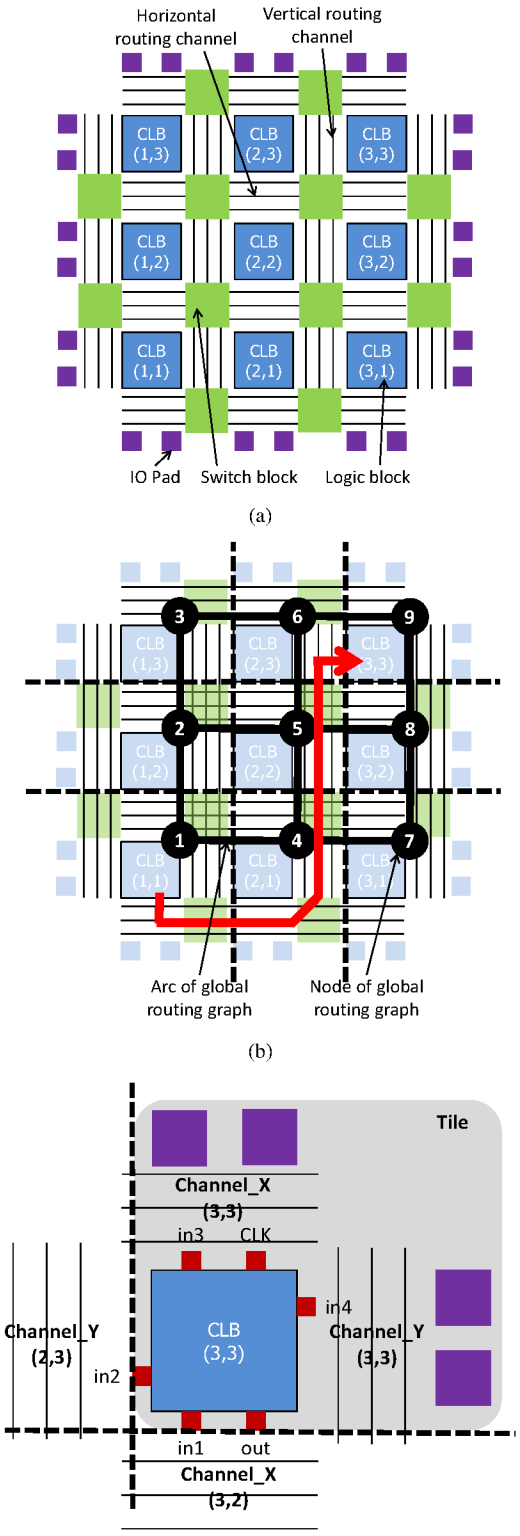

Fig. 2. (a) Typical FPGA architecture, (b) The global routing graph (GR-graph) is constructed by partitioning the architecture into a regular array of tiles and creating a node for each tile. Arcs connect nodes which correspond to adjacent tiles only. (c) Close-up of the tile which corresponds to the node 9 in the GR-graph. $in2$ of $CLB(3,3)$ is routed to from routing channel $Channel\_Y(2,3)$ located in the tile to the left.

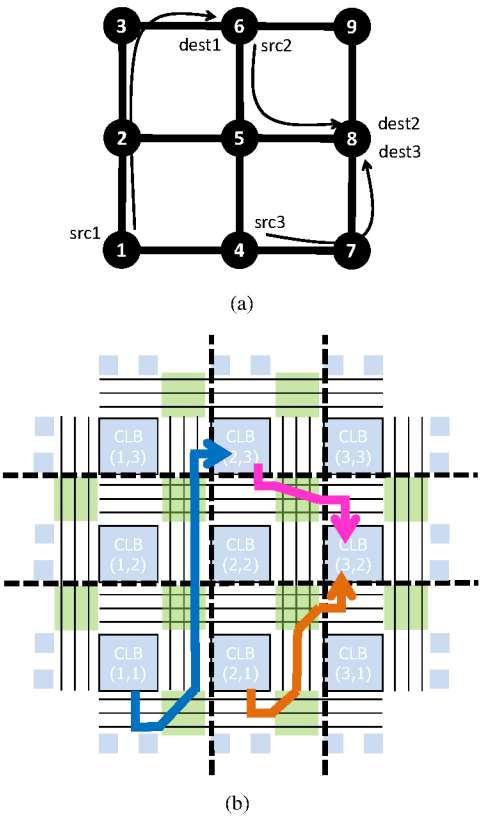

Fig. 3. (a) Example of MCF problem with three commodities shown as source-destination pairs and the global routing solution shown with arrows, (b) The final detailed routing which follows the paths indicated by the global routing solution.

Algorithm: Enhanced PathFinder Routing Algorithm

```
1: Phase 1: Global routing
2: Construct MCF problem for two-pin nets
3: Solve MCF problem using approx. algorithm to find global paths
4: Phase 2: Detailed routing
5: while (overused routing resources exist) do
6:     Process first nets with a number of pins between 4
7:     and 11, then nets with 2 and 3 pins, and finally nets
8:     with more than 12 pins
9:     for (each net, i) do
10:        Rip-up and reroute net i using PathFinder
11:        Modify cost calculations during wavefront expansions
12:        for two-pin nets to encourage detailed routes to
13:        follow preferred global paths
14:    end for
15: end while
```

Fig. 4. Pseudocode of the enhanced PathFinder routing algorithm.

following the global routing paths for two-pin nets or using the shortest paths for all other nets irrespective of whether routing resources are overused, this net ordering strategy favors final routing paths which are as close as possible to the initial shortest paths for nets with 2, 3, and greater than 11 pins while nets with a number of pins between 4 and 11 are *moved away* from overused resources just enough to make all routings legal (i.e., no routing resource overused).

B. Simulation Results

We have modified the PathFinder routing algorithm inside the VPR tool to integrate the techniques described in the previous sections. We applied these changes to both the routability- and timing-driven routing algorithms. We report simulation results collected for all twenty testcases of VPR 4.3 [10] and listed in Table I. These results are obtained for a channel-width equal to the minimum channel-width found with the reference[2] PathFinder for each testcase.

TABLE I
LIST OF INVESTIGATED TESTCASES. *NETS* REPRESENTS THE NUMBER OF NETS IN THE NETLIST OF EACH TESTCASE.

Testcase	Nets	Testcase	Nets
ex5p	1072	s298	1935
tseng	1099	bigkey	1936
apex4	1271	frisc	3576
misex3	1411	spla	3706
alu4	1536	elliptic	3735
diffeq	1561	pdc	4591
dsip	1599	ex1010	4608
seq	1791	s38417	6435
des	1847	s38584.1	6485
apex2	1916	clma	8445

First, we investigate the routability-driven routing algorithm. In this case, we expect the proposed enhanced PathFinder to reduce circuit delay because of the net reordering. We also expect for routability to be improved due to the global routing based cost alteration during wavefront expansions during the detailed routing step of two-pin nets. Indeed, the global routing of two-pin nets provide an initial routing solution for these nets that is closer to the actual final routing than what would otherwise the conventional initial routing provide. In other words, the initial routing solution (starting with the first iteration of the outer loop in Fig.4) starts with fewer overused routing resources and this in turn results in fewer routing iterations to reach a feasible routing solution for all nets. This is confirmed in Fig.5.a where we report the difference between the number of iterations required to reach a feasible routing solution when the enhanced PathFinder is used versus the case when the reference PathFinder is used. Fig.5.b shows the difference as percentage between the critical path delays achieved with the enhanced and reference PathFinder routers. We do not plot here the total wirelength, which on average remained the same in both cases.

In the second set of experiments, we investigate the timing-driven routing algorithm. The timing driven routing algorithm of VPR is also based on PathFinder but uses a cost function for wavefront expansions that combines congestion and Elmore delay cost components. Therefore, in this case, we expect the proposed enhanced PathFinder to have a lesser impact on routability but a larger impact on circuit delay. Fig.6 shows the results in this case where we can see that critical path delay is improved for several testcases while the number of routing iterations remains virtually the same on average.

[2]VPR tool is run using default values for all user-controlled parameters.

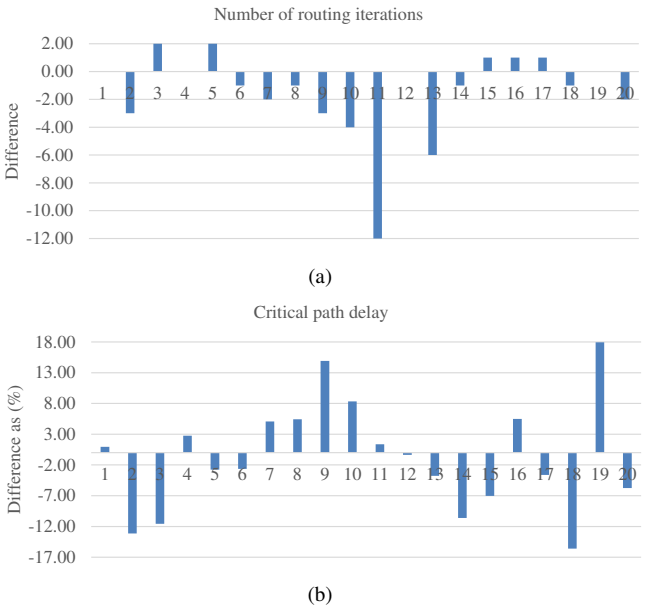

Fig. 5. (a) Difference between the number of routing iterations of routability-driven enhanced PathFinder and of the reference case for each of the twenty testcases. A negative value means the enhanced PathFinder finishes in fewer routing iterations. (b) Difference as percentage between the critical path delay obtained in the two cases.

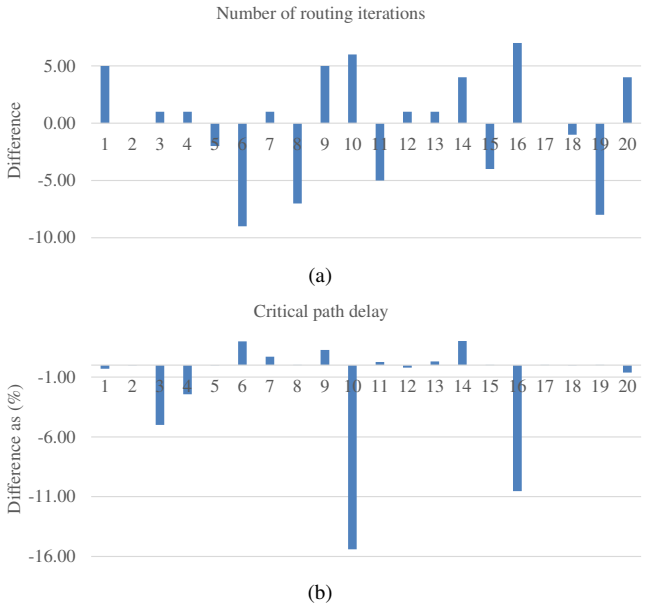

Fig. 6. (a) Difference between the number of routing iterations of timing-driven enhanced PathFinder and of the reference case for each of the twenty testcases. (b) Difference as percentage between the critical path delay obtained in the two cases.

VI. DISCUSSION AND FUTURE WORK

We notice that most testcases have several nets with a very large number of pins and they are always on the timing critical path. Practically, the pins of these nets are spread over the entire area of the FPGA. These cases are very difficult to improve in terms of wirelength and circuit delay. That is on one hand because the bounding box of such nets is already

very large (as a result of the placement step and cannot be reduced during routing) and on the other hand because the reference PathFinder is already finding good enough routing paths for these nets through its negotiated congestion rip-up and reroute process. This suggests that larger improvements could potentially be achieved through techniques that include logic restructuring to eliminate nets with very large number of pins and buffering.

In the routability-driven case, for most testcases the enhanced PathFinder reaches a feasible routing solution in fewer iterations. While a reduction in the the total number of iterations results into reduction of the computational runtime (hence this strategy can be seen as an alternative to [16] to reduce runtime), in this case, the reduction is overwhelmed by the runtime spent on constructing and solving the MCF problem.

The total computational runtime of all the routing steps is longer with about a quarter of the reference runtime for the small size circuits from Table I. This difference scales up with the circuit size primarily because the computational runtime of the proposed global routing step increases due to the increase of the MCF problem. This can be addressed by working with global routing graphs which are coarser: a node of the GR-graph can be associated, for instance, with clusters of four or more CLBs. Another idea is to implement hierarchical global routing, thereby speeding up the computational runtime. To further mitigate the net ordering problem, we can globally route nets with more than two pins as well. This can be done by transforming these nets into separate two-pin nets and then adding them to the MCF problem formulation. This transformation can be done by first using a Steiner minimal tree (SMT) algorithm for each net and then splitting the net at the Steiner points. Such investigations are left to future work.

VII. Conclusion

Aimed at addressing the net ordering problem for FPGA routing, we investigated two enhancement techniques for PathFinder, the most popular routing algorithm during the last two decades. The first technique changes the order in which nets are ripped-up and rerouted to give give higher priority to nets with two, three, and more than eleven pins. The second technique modifies the cost calculation during wave expansions of two-pin nets to take into account the global routing solution obtained by solving an equivalent multicommodity flow problem. Preliminary results show that these techniques can reduce the number of routing iterations while slightly improving the critical path delay in the case of routability-driven routing. In the case of timing-driven routing, these techniques result in improved critical path delay for several testcases. The total wirelength remains virtually the same in both cases. The source code of the enhanced PathFinder router is publicly available at [17].

Acknowledgment

This work was supported in part by the Department of Electrical and Computer Engineering at Marquette University.

References

[1] V. Betz and J. Rose, "VPR: a new packing, placement and routing tool for FPGA research," *FPL*, 1997.

[2] Vaughn Betz and Jonathan Rose, Architecture and CAD for Deep-submicron FPGAs, Kluwer Academic Publishers, 1999.

[3] J. Hu and S.S. Sapatnekar, "A survey on multi-net global routing for integrated circuits," *Integration: The VLSI Journal*, vol. 31, no. 1, pp. 1-49, Nov. 2001.

[4] C. Albrecht, A.B. Kahng, I.I. Mandoiu, and A. Zelikovsky, "Multicommodity flow algorithms for buffered global routing," *Approximation Algorithms and Metaheuristics, Chapman & Hall/CRC*, 2007.

[5] L. McMurchie and C. Ebeling, "PathFinder: a negotiation-based performance-driven router for FPGAs," *FPGA*, 1995.

[6] I. Haritaoglu and C. Ayakanat, "A global routing heuristic for FPGAs based on mean field annealing," *FPL*, 1994.

[7] Y.-W. Chang, D.F. Wong, and C. K. Wong, "FPGA global routing based on a new congestion metric," *ICCD*, 1995.

[8] S. Lee, Y. Cheon, and M.D.F. Wong, "A min-cost flow based detailed router for FPGAs," *ICCAD*, 2003.

[9] J. Barreiros and E. Costa, "Global routing for lookup-table based FPGAs using genetic algorithms," *FPL*, 2003.

[10] V. Betz, VPR 4.3, [Online]. Available: http://www.eecg.toronto.edu/~vaughn/vpr/vpr.html

[11] T. Kong, "A novel net weighting algorithm for timing-driven placement," *ICCAD*, 2002.

[12] Mokhtar S. Bazaraa, John J. Jarvis, and Hanif D. Sherali, Linear Programming and Network Flows, Wiley, 2009.

[13] Y. Hu, Y. Zhu, H. Chen, R. Graham, and C.-K. Cheng, "Communication latency aware low power NoC synthesis," *DAC*, 2006.

[14] C. Ababei, "Efficient congestion-oriented custom Network-on-Chip topology synthesis," *IEEE Int. Conf. on Reconfigurable Computing and FPGAs (ReConFig)*, 2010.

[15] G. Karakostas, "Faster approximation schemes for fractional multicommodity flow problems," *ACM/SIAM SODA*, 2002.

[16] M. Gort and J.H. Anderson, "Reducing FPGA router run-time through algorithm and architecture," *FPL*, 2011.

[17] Software downloads at MES Lab, Marquette University, 2014. [Online]. Available: http://dejazzer.com/software.html.