Iterative Routing Algorithm of Inter-FPGA Signals for Multi-FPGA Prototyping Platform

Mariem Turki[1], Zied Marrakchi[2], Habib Mehrez[1], and Mohamed Abid[3]

[1] Laboratoire d'Informatique de Paris 6
Universite de Pierre et Marie Curie, Paris France
[2] Flexras Technologies, Paris France
[3] CES Laboratory
Sfax University, Tunisia

Abstract. Over the last few years, multi-FPGA-based prototyping becomes necessary to test System On Chip designs. However, the most important constraint of the prototyping platform is the interconnection resources limitation between FPGAs. When the number of inter-FPGA signals is greater than the number of physical connections available on the prototyping board, signals are time-multiplexed which decreases the system frequency. We propose in this paper an advanced method to route all the signals with an optimized multiplexing ratio. Signals are grouped then routed using the intra-FPGA routing algorithm: Pathfinder. This algorithm is adapted to deal with the inter-FPGA routing problem. Many scenarios are proposed to obtain the most optimized results in terms of prototyping system frequency. Using this technique, the system frequency is improved by an average of 12.8%.

1 Introduction

With the ever increasing complexity of system on chip circuits, the software and hardware developers can no longer wait for the fabrication phase to test their designs[3]. Currently, it is estimated that 60 to 80 percent of an ASIC design is spent performing verification [13]. FPGA-based prototyping is an important step in the creation of the final product and it is the key to the success of marketing in time.

Because the silicon area overhead of FPGA versus ASIC technology has been measured to be about 40x [14], FPGA programming technology requires that an ASIC logic design be partitioned across multiple FPGA devices to achieve the necessary device logic capacity. The number of FPGAs depends on the size of the prototyped system, ranging from a few [4] up to 60 FPGAs [5].

In order to map the design into a multi-FPGA board, a partitioning tool decomposes the design into pieces that will fit within the logic resources of individual FPGA devices. For some systems, partitioning must be performed so that routing restrictions in terms of available FPGA pin count and system

P. Brisk, J.G. de Figueiredo Coutinho, P.C. Diniz (Eds.): ARC 2013, LNCS 7806, pp. 210–217, 2013.

topology are taken into account. Indeed, the number of I/Os is increasing for each new FPGA generation, but the ratio FPGA I/Os over FPGA logic capacity is decreasing. Thus, the number of signals which appear at the interface and which should be transmitted between FPGAs, is significantly higher than the number of available traces between those FPGAs.

The communication of interpartition signals between FPGAs is based on routing algorithms. In this paper, we propose a new approach to route all the inter-FPGA signals, based on signal multiplexing technique. To reach this goal, we use an iterative routing algorithm called Pathfinder [6]. This algorithm was used to route the intra-FPGA signals. We extend it for the inter-FPGA signals in order to obtain the best routing results.

The rest of the paper is organized as follows. Section 2 is dedicated to the related works which addressed this problem. In section 3 we present the iterative routing algorithm used initially to route the intra-FPGA signals. Section 4 explains the scenarios we propose to test the performance of the routing algorithm. These scenarios includes the inter-FPGA signal form and also the routing graph direction. In section 5 we describe the multiplexing IP that we use to transfer the multiplexed signals. section 6 is dedicated to the experimental results and to the evaluation of the the proposed methods. Finally, section 7 concludes the paper.

2 Related Works

To address the inter-FPGA signal routing problem, authors in [8] proposed heuristic algorithms to solve multiterminal routing signals in partial crossbar architectures. In [9], multiterminal signals are decomposed into two-terminal nets. Therefore, routing algorithm is applied to these nets.

Bab et al [1] introduced time multiplexing of I/O pins. Multiplexing means that multiple design signals are assembled and serialized through the same board connection and then de-multiplexed at the receiving FPGA. In [2], the authors proposed a new multiplexing approach based on the Integer Linear Programming. The main objective of this study is to select which signals must be multiplexed and those which must not. Using this technique, all signals are transmitted on each phase, but only those with updated values are considered. Since all the signals are transmitted in each phase, the number of slot per phase is very large, and the system frequency is decreased.

3 Inter-FPGA Signals Routing Strategy

To route inter-FPGA signals, it is necessary to find an algorithm that can assign, in an optimized manner, signals to the available resources. The technique mentioned in the section II uses constructive routing algorithm. This algorithm keeps track of the reserved and available physical connections between FPGAs. The router applies Dijkstra's shortest path algorithm [7] to determine the shortest path between the source and destination FPGAs. If the shortest path exists, the

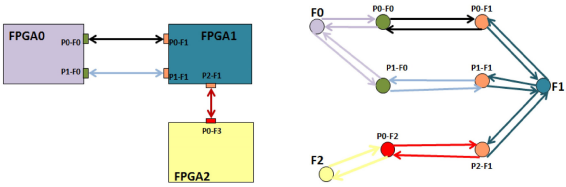

Fig. 1. Modelling routing resources as a routing graph

capacity of all used resources is decremented, then they can not be used to route the next signals. Else, router returns unsuccessfully. The main disadvantage of this method is: when a signal is already routed, it can not be rerouted to leave the routing resources currently used to another signal that has the greatest need for these resources. To avoid this problem, we route the inter-FPGA signals by an iterative routing algorithm. Among existing techniques, The Pathfinder routing algorithm seems to be best suited to our problem as it offers a compromise between performance and routability goals.

3.1 Routing Graph

Since we have chosen Pathfinder to route all inter-FPGA signals, our interest was about the modeling of the multi-FPGA board. Therefore, we chose to model all the routing resources by an oriented routing graph G(V, E). Like shown on Figure 1, the set of vertices V=$v_1, ...v_n$ in the graph represents the I/O pins of all FPGAs, but also, each FPGA is represented by a top vertex. The set of edges E=$e_1, ..., e_n$ represents all the inter-FPGA connections. An unidirectional connection is modeled by a directed edge while a bidirectional connection(for example between a vertex and a top vertex) is represented by two directed edges.

3.2 Routing Algorithm: Pathfinder

Pathfinder is used primarily for routing intra-FPGA signals. We adapt it to deal with the inter-FPGA signals. Pathfinder uses an iterative, negotiation-based approach to successfully route all the signals. During the first routing iteration, the signals are freely routed without paying attention to resource sharing. Individual signals are routed using Dijkstra's shortest path algorithm [7]. At the end of the first iteration, resources may be congested because multiple signals have used them. During subsequent iterations, the cost of using a resource is increased, based on the number of signals that share the resource, and the history of congestion on that resource. Thus, signals are forced to negotiate for routing resources. If a resource is highly congested, nets which can use lower congestion alternatives are forced to do so. On the other hand, if the alternatives are more congested than the resource, then a signal may still use that resource.

Observing the final routing results, we notice that inter-FPGA signals can be directly routed between source and destination FPGAs, or intermediate through-hops may be necessary.

4 Routing Algorithm Adaptation

Taking into account some problems to be detailed later, we adapt our routing approach to the new routing topology. In this section, we discuss the proposed solutions and the various changes we make.

4.1 Signal Direction Conflicts

The Pathfinder routing algorithm processes each signal independently. Each routing resource (node) should be shared by more than one signal. Signals that share the same resource are multiplexed together. As mentioned above, we model our architecture by a bidirectional routing graph. This causes direction conflicts since the signals sharing the same resources can have different directions.

Unidirectionnal Routing Graph. To avoid this problem, we apply the Pathfinder routing algorithm on a unidirectional graph. The idea is to assign a definite direction to all physical wires. In the routing graph, this is translated by a single edge between each pair of nodes.

Figure 2-(a) represents the routing flow on a unidirectionnal graph. The first generates the unidirectionnal graph depending on the number of inter-FPA signals between each pair. The number of physical wires that transmit direct (respectively indirect) signals between two FPGAs, is proportional to the number of direct (respectively indirect) signals between these two FPGAs. After calculating the multiplexing ratio, the capacity of all nodes is set to mux_ratio. Finally, Pathfinder routing algorithm tries to route all the signals. If a feasible solution exists, the mux_ratio parameter is decremented and the router tries to re-route the signals with the new value of mux_ratio. Otherwise, the router stops with the best solution found.

Bidirectionnal Routing Graph. The selection of the unidirectionnal wires proportionally to the number of signal between each pair of FPGA is not optimized at all. For this reason we keep the bidirectionnal graph and we combine signals into groups. Indeed, signals that have the same source and the same destinations are grouped together in "GSignals" and are considered as a single signal. Each GSignal contains a maximum of mux_ratio signals. Therefore, the capacity of all resources in the routing graph is set to 1. The bidirectional graph allows a better use for available routing wires of the multi-FPGAs prototyping board.Figure 2-(b) presents the steps to route inter-FPGA signals on a bidirectional routing graph. The first step creates the graph using two arcs of opposite direction to represent each physical wire. Next, the initial mux_ratio parameter is calculated the same way as in the unidirectional graph. This parameter determines the number of signals to be grouped together into one GSignal. The

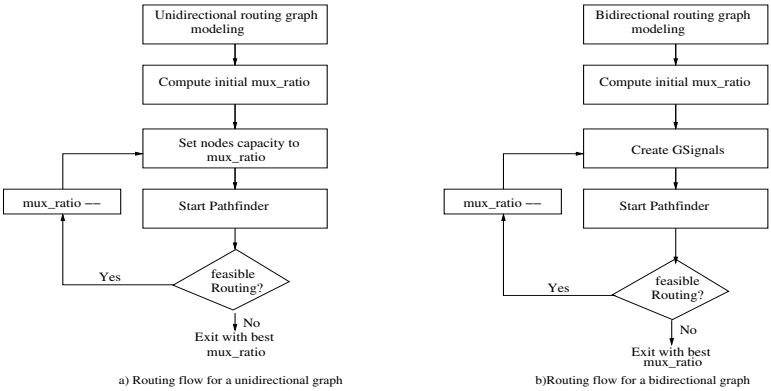

Fig. 2. Routing flows

pathfinder algorithm tries to route all the GSignals. Finally, the router retains the routing solution with the best mux_ratio.

This method avoids conflict management, since the Pathfinder algorithm prevents congestion; at the end of every iteration, no node is used by more than one group of signals or GSignals, which all have the same direction.

4.2 Signal Representation

For better routing results, we notice that the choice of signal form is essential with two possibilities to consider the signal shape: a multiterminal or a two-terminal signal.

Multiterminal Signal. The Pathfinder routing algorithm can route multiterminal nets. In fact, the algorithm starts by selecting the source and the list of all destinations. After routing the first one, Pathfinder moves to the next destination and so on. Although the routing of multiterminal signals can be the optimal solution considering the number of used I/O pins, the design is considered non flexible especially when grouping those signals into GSignals. Indeed, in some cases, signals with the same source and the same destinations are not numerous so that some GSignals do not contain the max number of signals, equal to mux_ratio.

Two-Terminal Signal. In order to make the design more flexible, we decompose the multiterminal signals into branches with one source each and only one destination. The Pathfinder routing algorithm tries to find separately a routing path to each branch.

Table 1. Comparison of routing strategies effects on prototyping system performance

Benchmark	scenario1			scenario2			scenario3			scenario4		
	mux_ratio	R_hop	Freq (MHz)	mux_ratio	R_hop	Freq (MHz)	mux_ratio	R_hop	Freq (MHz)	mux_ratio	R_hop	Freq (MHz)
Circuit A	12	2	17.85	15	2	16.66	4	2	20.83	4	1	26.31
Circuit B	18	3	13.88	24	2	14.7	4	3	17.24	7	1	23.8
Circuit C	24	3	12.82	44	2	11.36	11	3	15.15	11	1	21.73
Circuit D	50	3	9.61	50	2	10.63	15	3	14.28	20	1	18.51
Circuit E	119	6	4.9	116	4	5.55	57	2	9.8	56	4	8.33
Circuit F	160	3	4.67	168	3	4.5	68	3	8.19	68	1	9.8
Circuit G	220	5	3.4	256	1	3.44	89	2	7.46	86	3	7.14

Table 2. Comparison between OAR and NCR strategies on system performance

Benchmarks	OAR			NCR			Gain
	R_hop	mux_ratio	Freq(MHz)	R_hop	mux_ratio	Freq(MHz)	
CPU50_occ30	0	9	29.41	0	9	29.41	0%
CPU125_occ50	2	16	16.66	1	16	20	20.04%
CPU150_occ30	3	24	12.82	1	29	15.62	21.84%
CPU150_occ50	2	51	10.41	1	51	11.62	11.65%
CPU375_occ80	2	51	10.41	1	51	11.62	11.65%
CPU375_occ85	2	79	8.06	2	69	8.77	8.8%
CPU700_occ80	2	134	5.61	2	109	6.49	15.68%

5 Experimental Results

We use our benchmark generator [11] to generate several synthetic designs. The targeted multi-FPGA prototyping board we use for the experiments is a DNV6F6PCIe from the DINI group [12]. The inter-FPGA clock frequency is set to 500MHz. To map the designs into this board, we use the WASGA partitioning flow provided by Flexras Technologies [10]. WASGA partitions the designs and outputs the list of inter-FPGA signals that shoud be routed. After routing these signals, WASGA generates a netlist for each FPGA. The resulting netlists are re-entered into the FPGA flow to execute the place and route and the bitstream generation individulally for each FPGA.

Table 1 shows the results for each routing scenarios described in section 4. These scenarios are defined depending on the signal shape and the routing graph.

- In the scenario 1, multiterminal signals are routed on a unidirectional routing graph.
- In scenario 2, two-terminal branchs are routed into a unidirectional routing graph.
- In scenario 3, Signals are grouped into GSignals and routed into a bidirectional graph.
- In scenario 4, Branches are combined into groups and routed into a bidirectional graph.

In this experiment, we used benchmarks where 70% of signals are multiterminal ones. Results show that routing on a bidirectional graph gives much better results since the router is more free to select the routing path. On the other hand, routing multiterminal signals is not always optimized even if the mux_ratio of scenario 3 is sometimes less than the one of scenario 4, but using more routing hop penalizes the system frequency.

Since we have demonstrated that Senario 4 gives usually the best results, we apply Pathfinder and the obstacle avoidance routing algorithms to route inter-FPGA signals, all with one source and one destination (branch) and grouped into GSignals. Table 2 shows the results of comparison. OAR means Obstacle Avoidance Routing and NCR refers to Negotiated Congestion Routing. Results show the important impact of the NCR iterative routing and its efficiency to improve system performance. The frequency is increased on average by 12.8% and the impact of NCR is important for highly congested partitioning results. In fact thanks to its iterative aspect, it avoids easily local minima and reduces the path length from a source FPGA to a destination FPGA. In addition, it leads to a good tradeoff between maximum multiplexing ratio and routing hops.

6 Conclusion

Prototyping is no longer optional due to the cost of chips and difficulty to simulate huge designs. To get a design for prototype more efficient, the highest frequency should be reached. The system frequency depends on the way the inter-FPGA signals are routed. n this paper, we presented our approach to route these inter-FPGA signals. We extend the Pathfinder routing algorithm to deal with the inter-FPGA signals. These signals are grouped into GSignals where each one has 1 source and only 1 destination. Compared to common obstacle avoidance algorithms, we obtain a significant prototyping system frequency improvement of 12.8%.

References

1. Tessier, R., et al.: The virtual Wires Emulation System: A Gate-Efficient ASIC Prototyping Environement. In: International Workshop on Field-Programmable Gate Array. ACM, Berkeley (February 1994)
2. Inagi, M., Takashima, Y., Nakamura, Y.: Globally optimal time-multiplexing in inter-FPGA connections for accelerating multi-FPGA systems. In: Proc. FPL, pp. 212–217 (2009)
3. Huang, C., Yin, Y., Hsu, C.: SoC hw/sw verification and validation. In: Proc. of the 16th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 297–300 (2011)
4. Krupnova, H.: Mapping multi-million gate SoCs on FPGAs: industrial methodology and experience. In: Proc. of Design, Automation and Test in Europe Conference and Exhibition, vol. 2, pp. 1236–1241 (2004)

5. Asaad, S., Bellofatto, R., Brezzo, B., Haymes, C., Kapur, M., Parker, B., Roewer, T., Saha, P., Takken, T., Tierno, J.: A cycle-accurate, cycle reproducible multi-FPGA system for accelerating multi-core processor simulation. In: Proc. of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 153–162 (2012)
6. McMurchie, L., Ebeling, C.: PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs. In: International Workshop on Field Programmable Gate Array (1995)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Cambridge, Massachusetts London, England (2001)
8. Ejnioui, A., Ranganathan, N.: Multiterminal net routing for partial crossbar-based multi-FPGA systems. IEEE Trans. Very Large Scale Integr (VLSI) Syst. 11(1), 71–78 (2003)
9. Mak, W., Wong, D.: Board-level multiterminal net routing for FPGA-based logic emulation. ACM Trans. Design Automation of Electron. Syst. 2, 151–157 (1997)
10. `http://www.flexras.com`
11. Turki, M., Marrakchi, Z., Mehrez, H., Abid, M.: Towards Synthetic Benchmarks Generator for CAD Tool Evaluation. In: 8^{th} Confrence on Ph.D. Research in Microelectronics and Electronics (PRIME) (2012)
12. `http://www.dinigroup.com/new/products.php`
13. Santarini, M.: ASIC prototyping: Make versus buy. EDN (November 21, 2005)
14. Kuon, I., Rose, J.: Measuring the gap between FPGAs and ASICs. In: International Symposium on Field-Programmable Gate Array (February 2006)