Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2013, Article ID 853510, 12 pages
http://dx.doi.org/10.1155/2013/853510

Research Article

Frequency Optimization Objective during System Prototyping on Multi-FPGA Platform

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

[1] *LIP6, UPMC, 75005 Paris, France*
[2] *FLEXRAS Technologies, 93521 Saint-Denis Cedex, France*
[3] *CES Lab, ENIS, 3038 Sfax, Tunisia*

Correspondence should be addressed to Mariem Turki; mariem.turki@lip6.fr

Received 30 April 2013; Accepted 16 October 2013

Academic Editor: Michael Hübner

Multi-FPGA hardware prototyping is becoming increasingly important in the system on chip design cycle. However, after partitioning the design on the multi-FPGA platform, the number of inter-FPGA signals is greater than the number of physical connections available on the prototyping board. Therefore, these signals should be time-multiplexed which lowers the system frequency. The way in which the design is partitioned affects the number of inter-FPGA signals. In this work, we propose a set of constraints to be taken into account during the partitioning task. Then, the resulting inter-FPGA signals are routed with an iterative routing algorithm in order to obtain the best multiplexing ratio. Indeed, signals are grouped and 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% compared to constructive routing algorithm.

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 [1]. Currently, it is estimated that 60 to 80 percent of an ASIC design is spent in performing verification [2].

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. The key advantage of FPGA-based prototyping is the ability to run at high speed (sometimes at almost real-time speed) a cycle-accurate, bit-accurate model of the SoC [3]. The availability of automatic FPGA mapping tools has streamlined the design conversion process, making the path from ASIC design to FPGA implementation more straightforward.

When the logic capacity of a single FPGA is less than the size of the design under test, a multi-FPGA platform is used to map the entire design. Because the silicon area overhead of FPGA versus ASIC technology has been measured to be about 40x [4], 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 prototyping system, ranging from a few [5] up to 60 FPGAs [6].

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. Partitioning is often performed to minimize required inter-FPGA interconnect, control system-wide critical path delay and localize memory access. For some systems, partitioning must be performed so that routing restrictions in terms of available FPGA pin count and system topology are taken into account. These constraints are considered in order to manage the large timing delays in inter-FPGA communications compared to intra-FPGA ones and also to cope with the limited bandwidth problem between FPGAs, which is due to the limited I/Os per FPGA. Indeed, the number of I/Os is increasing for each new FPGA generation, but the ratio of FPGA I/Os over FPGA logic capacity is decreasing. Thus, even though the logic capacity of the multi-FPGA board is sufficient to map the complete design, 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. The most used routing algorithm involves the determination of the shortest feasible path between FPGAs, using available board interconnect resources for each cut signal [7]. This approach is not recursive and leads inevitably to a blockage.

In this paper, we propose a set of constraints to be considered during the partitioning task. These constraints are intended to get the best results in terms of the number of cut signals and the critical path optimization. We propose also a new approach to route the resulted inter-FPGA signals, based on signal multiplexing technique. To reach this goal, we use an iterative routing algorithm, called Pathfinder [8]. 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 this paper is organized as follows. Section 2 is dedicated to the different techniques used in the state of the art to route the inter-FPGA signals. Section 3 describes the different steps of the design prototyping flow. In Section 4, we present the proposed routing algorithm which is used initially to route the intra-FPGA signals. Section 5 explains the scenarios we propose to test the performance of the routing algorithm. These scenarios include the inter-FPGA signal form and also the routing graph direction. In Section 6, we describe the multiplexing IP that we use to transfer the multiplexed signals. Section 7 is dedicated to the experimental results and to the evaluation of the proposed methods. Finally, Section 8 concludes the paper.

2. Related Works

To address the inter-FPGA signals routing problem, the authors in [9, 10] proposed heuristic algorithms to solve multiterminal routing signals in partial crossbar architectures. In [11, 12], multiterminal signals are decomposed into two-terminal nets. Therefore, routing algorithm is applied to these nets.

In this paper, our goal is to find the best signal shape which gives the best routing results. For this reason, many scenarios are applied with the proposed routing graph in order to get the best system frequency.

To remedy the number of pin limitations, Babb et al. [13, 14] introduced time multiplexing of I/O pins. Multiplexing means that multiple design signals are assembled and serialized through the same board connection and then demultiplexed at the receiving FPGA. This technique increases dramatically the available inter-FPGA communication bandwidth. On the other hand, it makes the prototyping system much slower since the system clock period is composed of several phases. Each phase contains a number of slots. Consequently, in each phase, the selected signals are transmitted, each in a slot, between a pair of FPGAs. Signals are selected based on their criticality which is calculated depending on the logic dependency analysis. A signal is selected if all signals it depends upon have been routed in previous phases. The router then uses the shortest path analysis with a cost function based on pin utilisation to route as many selected signals as possible, routing the most critical signals first. Any selected signals which cannot be routed are delayed to the next phase. In this technique, all the signals are multiplexed, without promoting the signals on the critical path. Some critical signals may not be multiplexed to obtain a better performance in terms of system frequency. Another disadvantage related to the combinational loops is that any unpredictable delay of an inter-FPGA signal causes the transmission of nonupdated values and then system errors. Even though such a sophisticated approach may realize faster verification speed, it decreases the reliability of circuit verification which is the most critical issue of circuit verification.

In [15, 16], 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 slots per phase increases and the system frequency is decreased. This technique, as the one in [9, 10], uses a constructive routing algorithm which is not optimized. In fact, when a signal is already routed, it cannot be rerouted to leave the routing resources currently used to another signal that has the greatest need for these resources. This disadvantage will be solved using an iterative routing algorithm as proposed in this study. In fact, the objective of our iterative approach is that all the signals negotiate the use of the routing resources. Each physical wire will be used by the signal which has the biggest need to this resource. This negotiation will be done through several iterations to solve all the conflicts, unlike the constructive routing algorithm which is done only by one iteration.

3. Prototyping Flow

To prototype an ASIC design into a multi-FPGA platform, the input circuit is transformed into multi-FPGA configuration bitstream to be downloaded onto the prototyping board. Figure 1 presents the prototyping flow.

3.1. Logic Synthesis. The HDL description files of the prototyping architecture are mapped onto the target library of FPGA primitives. In this paper, the benchmarks are synthesized with the Synplify industrial tool [18]. The output of this task is a postsynthesis Verilog netlist.

3.2. Partitioning. After mapping the netlist onto the target technology, it is divided into partitions; each can fit into a single target FPGA. The partitioner performs K-way partitioning with multiobjective function. The partitioning step is very critical since it has a significant impact on the performance of the prototyping system. In this study, we use the Wasga partitioning tool of Flexras technologies [19]. For

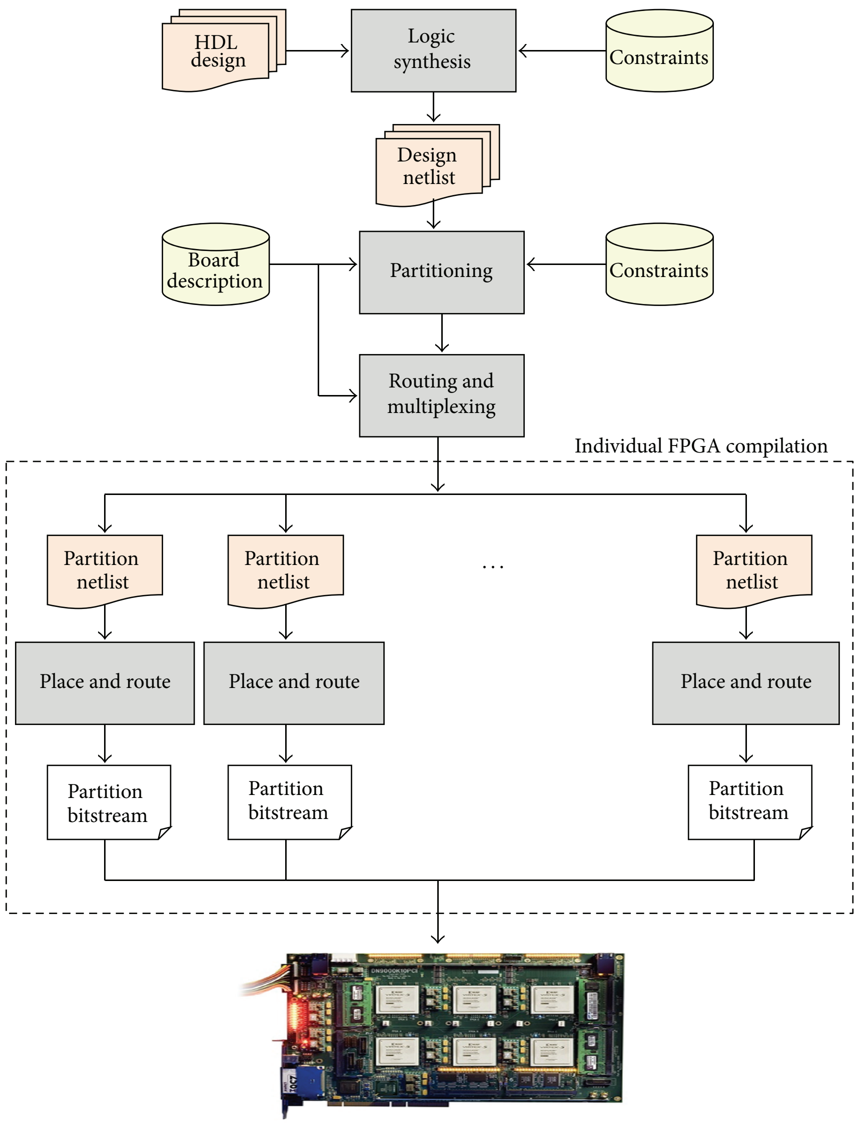


Figure 1: Prototyping flow.

this tool, we set some constraints in order to have a good trade-off between the following criteria.

(a) Minimize the Number of Cut Signals. For big designs, it is difficult, if not impossible, to find a partitioning solution which meets the constraint related to the number of physical connections between FPGAs. As will be explained subsequently, the solution is to make a postpartitioning process allowing a number of signals to share the same physical wire in different time fractions. The insertion of these multiplexers increases the delays on combinatorial paths. These delays are correlated to the number of multiplexed signals (multiplexing ratio). Thus, the main goal of the partitioner is to reduce the number of the cut signals in order to get the lowest rate of multiplexing.

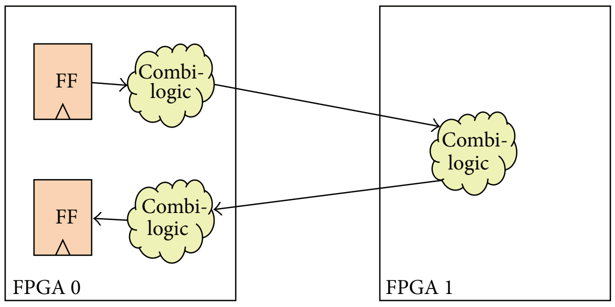
On the other side, the ratio between the number of cut signals and the number of available wires should be balanced between all pairs of FPGA. Therefore, the objective of the partitioning tool is to minimize the C_p parameter presented in

$$C_p = \sqrt{\sum_{p=0}^{N} \left(\frac{S_p}{T_p}\right)^2}, \quad (1)$$

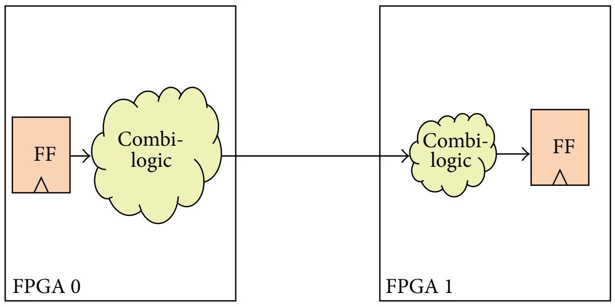
with N being the number of FPGA pairs in the prototyping platform, S_p being the number of signals between the pair p of FPGAs, and T_p being the number of available tracks between the same pair.

Finally, the partitioner aims to provide guidance about the signals which should not be multiplexed since they affect the critical path.

(b) Combinatorial Paths. The system frequency is imposed by the delay of the longest combinatorial path (between two registers). The delay on a combinatorial path is strongly correlated with the number of times a path crosses the border



(a) Partitioning solution with cut signals = 2, combi-hop = 2



(b) Partitioning solution with cut signals = 1, combi-hop = 1

Figure 2: Combinatorial hop example.

of an FPGA, called combinatorial hop. This is because the transmission through inter-FPGA connection is much slower than the one inside the FPGA. Therefore, it is important to absorb the signals belonging to the critical combinatorial paths. In Figure 2(a), the number of combinatorial hops is equal to 2, and the number of cut signals is equal to 2. If the partitioner identifies the best module to move, the partitioning solution will be improved since the number of inter-FPGA signals and the number of combinatorial hops will be reduced as shown in Figure 2(b).

(c) Logical Resources Limitation. The number of logical resources in the FPGA circuits is limited. During the partitioning, an occupancy rate constraint is set, so the partitioning tool must take into account this number and try to make a partitioning solution which meets the available resources. These resources are heterogeneous since they include different types (LUT, Ram, DSP, etc.). The occupancy rate should consider the additional logical area which will be occupied by the multiplexing IPs after the inter-FPGA routing tasks.

Unlike most of commercial tools, the partitioning tool used in our experiments operates on synthesized netlists which gives accurate information about the size of the design so it can meet the available logical resources of each FPGA.

3.3. Routing and Multiplexing. The system clock is the clock of the logic design being prototyped. The system clock period is divided into a number of slots as shown in Figure 3. Each signal is transmitted between a pair of FPGA within one slot period. These slots are controlled by an I/O clock which is faster than that of the system in order to transfer all the signals within one system clock period.

The system clock period is given by the following equation:

$$T_{\text{SYS\_CLK}} = \text{settle\_start} + \text{comm\_delay} + \text{settle\_end}. \qquad (2)$$

Settle_start and settle_end correspond to the intradelay of propagation inside the source and destination FPGA, respectively. During the intra-FPGA place and route tasks, we define a multicycle path constraint to set the intra-delay propagation to 3 times the intercommunication period; that is, $3 * T_{\text{IO}_{\text{clk}}}$ in order to relax the timing constraint inside each FPGA. The comm_delay is the delay of the inter-FPGA communication. This delay should be reduced in order to optimize the system frequency. The communication delay is represented by the following expression:

$$\text{Comm\_delay} = T_{\text{mux}} + T_{\text{routing\_hop}} + T_{\text{latencies}}. \qquad (3)$$

T_{mux} is the amount of delay spent to transfer all signals via the same physical wire and it is proportional to the multiplexing ratio. The $T_{\text{routing\_hop}}$ is the delay spent to cross all the routing hops. In fact, the number of routing hops is the number of FPGAs to cross to route a signal between the source and the destination. Finally, $T_{\text{latencies}}$ is the latency of the SERDES modules.

In order to reduce the multiplexing ratio, the effort should be spent on the routing task. Indeed, using an appropriate routing algorithm, the router can find the optimized solution related to the given constraints. As shown in Figure 4, the router takes as input the architecture of the prototyping platform, the list of cut signals to be routed, and the initial mux_ratio parameter which is the number of inter-FPGA signals to be transmitted through the same physical wire. This parameter is calculated as the max of the multiplexing ratio of all the FPGA pairs. The mux_ratio of one FPGA pair is the ratio between the number of signals and the number of connection wires between these two FPGAs.

Figure 4 shows the proposed flow to reduce the multiplexing ratio. Depending on the given inputs, the router tries to route all inter-FPGA signals by meeting the mux_ratio calculated initially. If a feasible solution exists, the mux_ratio is decremented and the router attempts to find another routing solution with the new mux_ratio. Otherwise, the router exits with the best obtained multiplexing ratio.

3.4. FPGA Place and Route. Once the routing is achieved, the multiplexing IPs are inserted on the source and destination FPGAs to ensure the inter-FPGA signals transmission in the corresponding time slots. One netlist is generated for each FPGA. Each netlist must be processed with FPGA specific automated place and route software to generate configuration bitstreams.

4. 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

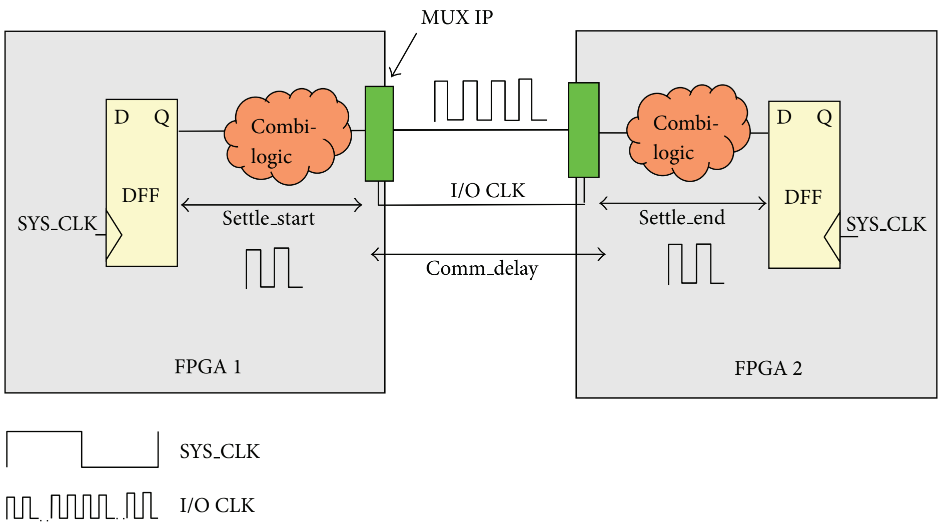


Figure 3: Clocking framework.

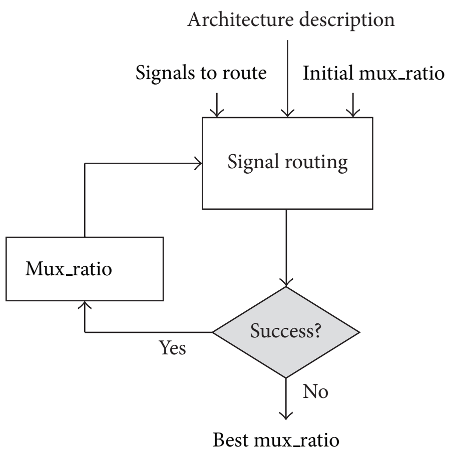


Figure 4: Mux_ratio optimization based on iterative approach.

available resources. The techniques mentioned in Section 2 use constructive routing algorithm. This algorithm keeps the track of the reserved and available physical connections between FPGAs. The router applies Dijkstra's shortest path algorithm [20] to determine the shortest path between the source and destination FPGAs. If the shortest path exists, the capacity of all used resources is decremented; then, they cannot be used to route the next signals. Otherwise, router returns unsuccessfully. The main disadvantage of this method is its irreversibility. Indeed, when a signal is already routed, it cannot be rerouted to leave the routing resources currently used to another signal that has the greatest need for these resources. In the example of Figure 5, signals are routed randomly. If the signal S1 is first routed through FPGA1, then S2 cannot be routed since the wire between FPGA1 and FPGA2 is used by S1. In this case, the design is considered nonroutable. 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.

4.1. Routing Graph. Since we have chosen Pathfinder to route all inter-FPGA signals, our interest was about the modelling of the multi-FPGA board. Therefore, we chose to model all the routing resources by an oriented routing graph $G(V, E)$. The set of vertices, $V = v_1, \dots, v_n$, in the graph represents the I/O pins of all FPGAs, and each FPGA is represented by a top vertex. The set of edges, $E = e_1, \dots, e_n$, represents all the inter-FPGA connections. An unidirectional connection is modelled by a directed edge, while a bidirectional connection (e.g., between a vertex and a top vertex) is represented by two directed edges.

Figure 6 presents a routing graph of a three-FPGA-based platform.

4.2. Routing Algorithm: Pathfinder. Pathfinder is used primarily for routing intra-FPGA signals. We adapt it to deal with the inter-FPGA signals [21]. Pathfinder uses an iterative, negotiation-based approach to successfully route all the signals. The routing problem for a given signal is to find a directed tree embedded in G that connects the source of the signal to each of its FPGA destinations. 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 [20]. 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.

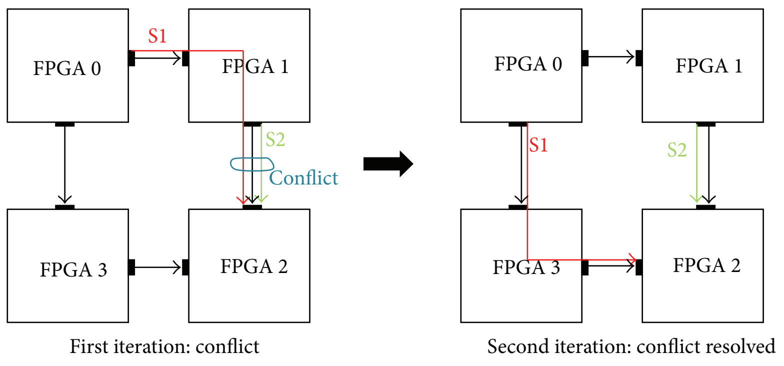

FPGA 0
FPGA 1
FPGA 3
FPGA 2
S1
S2
Conflict
First iteration: conflict
Second iteration: conflict resolved

Figure 5: Conflict resolution by an iterative routing algorithm.

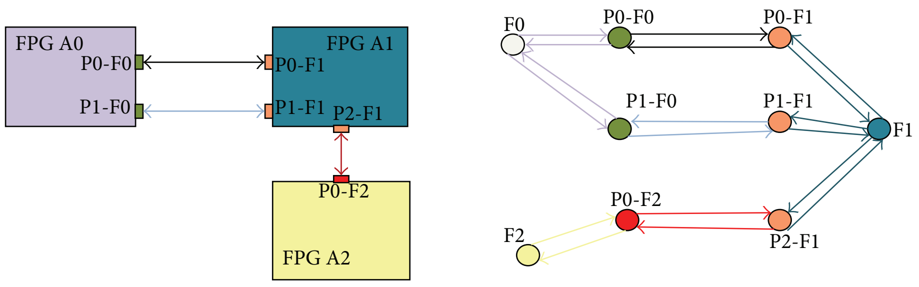


Figure 6: Modelling multi-FPGA platform as routing graph.

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.

5. 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.

5.1. Convention. All FPGAs on the prototyping board are indexed sequentially, starting at 0. We say that a signal has a direct direction if the index of the FPGA source is lower than its FPGA destination. Signal with indirect direction is the signal which is directed towards the opposite.

5.2. Signal Direction Conflicts. The Pathfinder routing algorithm processes each signal independently. Each routing resource (node) may 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.

5.2.1. Unidirectional Routing Graph. To avoid direction conflicts, we apply the Pathfinder routing algorithm on a unidirectional graph. The idea is to assign, according to criteria to be detailed later, a definite direction to all physical wires. In the routing graph, this is translated by a single edge between each pair of nodes.

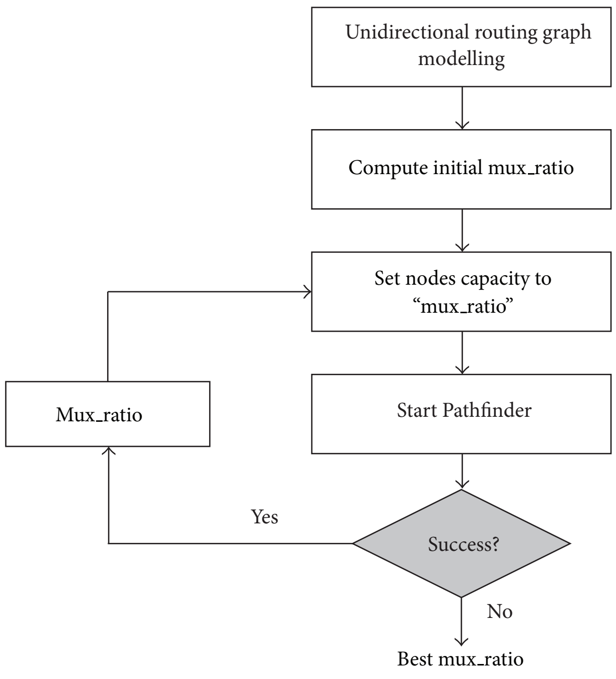

Unidirectional routing graph modelling
Compute initial mux_ratio
Set nodes capacity to "mux_ratio"
Start Pathfinder
Mux_ratio
Success?
Yes
No
Best mux_ratio

Figure 7: Routing flow on unidirectional graph.

Figure 7 represents the routing flow on a unidirectional graph. The first step generates the unidirectional graph depending on the number of inter-FPGA signals between each pair. The number of physical wires that transmit direct (resp., indirect) signals between two FPGAs is proportional to the number of direct (resp., indirect) signals between these two FPGAs. The following equation represents the number of physical wires in a given direction:

$$\mathrm{NB}_{\mathrm{wires}[f_i \to f_j]} = \frac{\mathrm{Sig}_{[f_i \to f_j]}}{\mathrm{Sig}_{[f_i, f_j]}} * \mathrm{NB}_{\mathrm{wires}[f_i, f_j]}. \quad (4)$$

$\mathrm{Sig}_{[f_i \to f_j]}$ and $\mathrm{Sig}_{[f_i, f_j]}$ are, respectively, the number of direct signals between FPGA_i and FPGA_j and the total number of signals between the same pair. $\mathrm{NB}_{\mathrm{wires}}[f_i, f_j]$ is the total number of available physical wires between FPGA_i and FPGA_j.

In the example of Figure 8, the number of direct wires is set to 3, and the number of indirect wires is set to 2. The second step consists in computing the initial mux_ratio. This parameter is calculated as follows:

$$\text{mux\_ratio} = \mathrm{Max}_{f1 \to f2 \in \mathrm{FPGAs}} \frac{\mathrm{Sig}_{f1 \to f2}}{\mathrm{Wires}_{f1 \to f2}}. \quad (5)$$

The maximum mux_ratio of all the FPGA pairs is the ratio between the number of signals and the number of available physical wires between each pair.

After calculating the multiplexing ratio, the capacity of all nodes is set to mux_ratio. Then, Pathfinder routing algorithm tries to find a feasible solution in which all signals should be routed and each node should not be shared by more than "mux_ratio" signals. If these two constraints are met, the mux_ratio parameter is decremented and the router tries to find a feasible solution with the new value of mux_ratio. Otherwise, the router exits with the best solution found.

5.2.2. Bidirectional Routing Graph. The selection of the unidirectional wires proportional to the number of signals between each pair of FPGA is not an optimized decision. For this reason, we keep the bidirectional graph and we assemble 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-FPGA prototyping board.

Figure 9 presents the steps to route inter-FPGA signals on a bidirectional routing graph. The first step creates the graph using two edges of opposite directions to represent each physical wire. Next, the initial mux_ratio parameter is calculated in the same way as in the unidirectional graph. This parameter determines the number of signals to be grouped together into one GSignal.

After running the Pathfinder algorithm to route the GSignals, all GSignals should be routed and each node should be used by only one GSignal. 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.

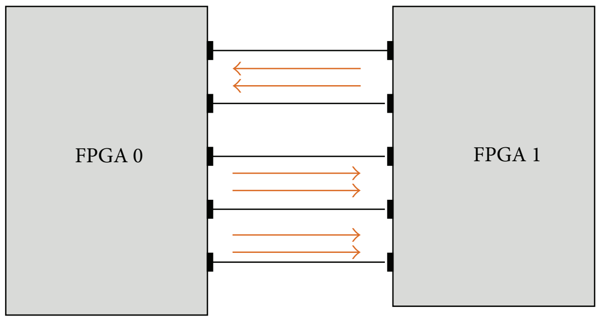


FIGURE 8: Unidirectional wires selection proportional to the number of signals.

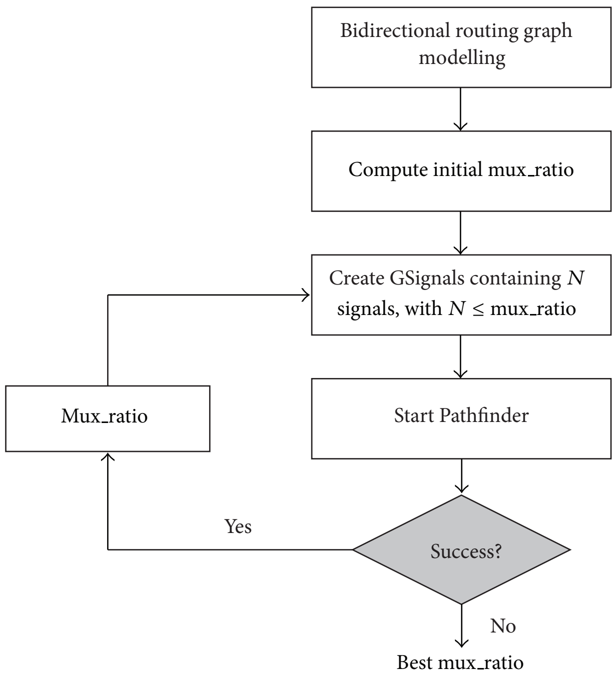

Bidirectional routing graph modelling
Compute initial mux_ratio
Create GSignals containing N signals, with N ≤ mux_ratio
Mux_ratio
Start Pathfinder
Yes
Success?
No
Best mux_ratio

FIGURE 9: Routing flow on bidirectional graph.

However, this method is not fully optimized. Indeed, from a source "S" to destination "D," the number N of existing signals can be much below the maximum number of signals allowed in a GSignal, which is equal to mux_ratio. Consequently, since the capacity of the nodes is set to 1, in a path between "S" and "D," only N signals are routed, which means a bad routing resource utilization.

5.3. Signal Modelling. 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.

5.3.1. Multiterminal Signal. After partitioning the prototyping design, the next step routes all nets from the part containing the driver of this net to all parts containing destinations.

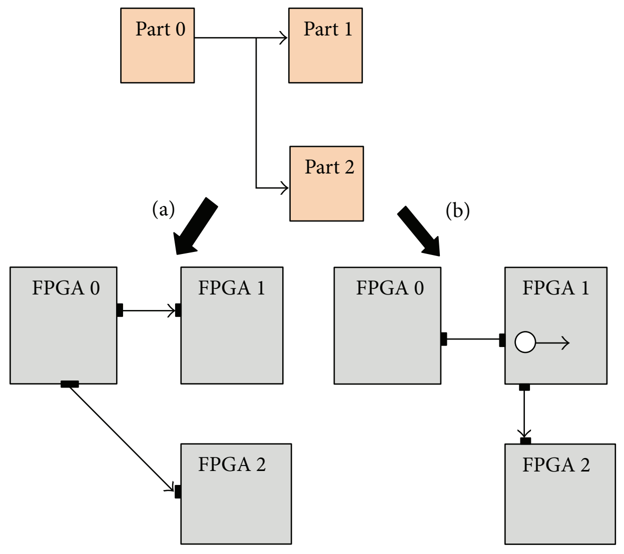


Figure 10: Routing solutions of multiterminal net.

Actually, a signal can have more than one destination. 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. Such a signal can be routed in 2 different ways as shown in Figure 10. The first solution is to use 2 different paths to route the source to its 2 destinations. Indeed, after routing the first destination via path 1, the router selects the second destination and tries to route it. An uncongested path can be found via a second one that does not intersect with path 1. This solution is represented by Figure 10(a). The second solution is to route a destination using a part of path used to route an already routed destination. It means that a destination can be reached from the last routed one as shown in Figure 10(b).

Although the routing of multiterminal signals can be the optimal solution, considering the number of used I/O pins, the design is considered unflexible, especially when grouping those signals into GSignals. Indeed, in some cases, signals with the same source and the same destinations are not numerous; consequently, some GSignals do not contain the max number of signals, equal to mux_ratio.

5.3.2. Two-Terminal Signal. In order to make the design more flexible, we decompose the multiterminal signals into branches, each with one source and only one destination. The Pathfinder routing algorithm tries to find separately a routing path for each branch. With this decomposition, only the solution shown in Figure 10(a) is feasible.

6. Multiplexing IP

The approach described above determines which signals to be multiplexed together. These signals are transmitted through the same physical wire and transferred using 2 multiplexing IP placed in the sending and receiving FPGAs, as shown in Figure 11. To ensure the inter-FPGA communications, dedicated output parallel-to-serial converters (OSERDES) and input serial-to-parallel converters (ISERDES) are instantiated in the sending and receiving FPGAs. The low-voltage differential signalling (LVDS) is used to transfer the data between SERDES converters. The LVDS is a signalling standard providing high-speed data transfers.

When the number of cut signals exceeds the number of available I/O pins, the partitioning tool inserts 4-bit wide SERDES converters. Nevertheless, if the number of cut signals is not a multiple of 4, some OSERDES inputs (resp., ISERDES outputs) can be left unconnected. The maximum number of signals transmitted between an ISERDES/OSERDES pair is defined as mux_ratio ($2 \leq \text{mux\_ratio} \leq 4$).

In highly connected designs, the number of signals can still exceed the capacity of transmission between a pair of FPGA, even after implementing the SERDES converters. In this case, n-bit wide MUXs (resp., DEMUXs) are added at the input of the OSERDES (resp., at the output of the ISERDES). The number n equals the number of 4-bit data words to be sent. When the mux_ratio is less or equal to 4, then $n = 1$. On the other side, if mux_ratio ≥ 5, then $n \geq 2$.

The combination of one 4-bit wide SERDES with one n-bit wide MUXs/DEMUXs constitutes the multiplexing IP. This IP manages the inter-FPGA communication by sending the data, as well as a 4-bit start pattern (for the inter-FPGA synchronisation) and a 4-bit checksum (to verify the integrity of the transmitted data). Since 2 inter-FPGA clock cycles are required to send a 4-bit word, then $2 + 2 + 2 * n$ are needed to send the start pattern, the checksum, and the n 4-bit data. If we consider the SERDES converters latencies as well as the IP latency, $12 + 2 * n$ cycles are needed to complete the communication between a pair of FPGA.

On the other hand, some signals are routed through hops since the direct path from the source to the destination does not exist. So, when a signal is routed through one or more routing hops, we insert 5 registers in the netlist of each routing hop in order to recover the 4-bit data before sending it to the next FPGA on the routing path. Therefore, 5 cycles are needed to cross a hop.

According to (3), the communication delay is equal to

$$\text{Comm\_delay} = \text{NB}_{R\_\text{hop}} * 5 + 12 + \frac{\text{mux\_ratio}}{2}. \qquad (6)$$

Since the comm_delay causes the biggest delay, we neglect the effect of the intradelays into the sending and the receiving FPGAs defined in (2). Therefore, the system frequency is represented in

$$\text{Sys\_freq} = \frac{\text{I/O frequency}}{\text{NB}_{R\_\text{hop}} * 5 + 12 + \text{mux\_ratio}/2}. \qquad (7)$$

7. Experimental Results

We use the benchmark generator [22] to generate several synthetic designs. The generated benchmarks are hierarchical since the partitioner operates on high levels of hierarchy in order to reduce the partitioning runtime and the number

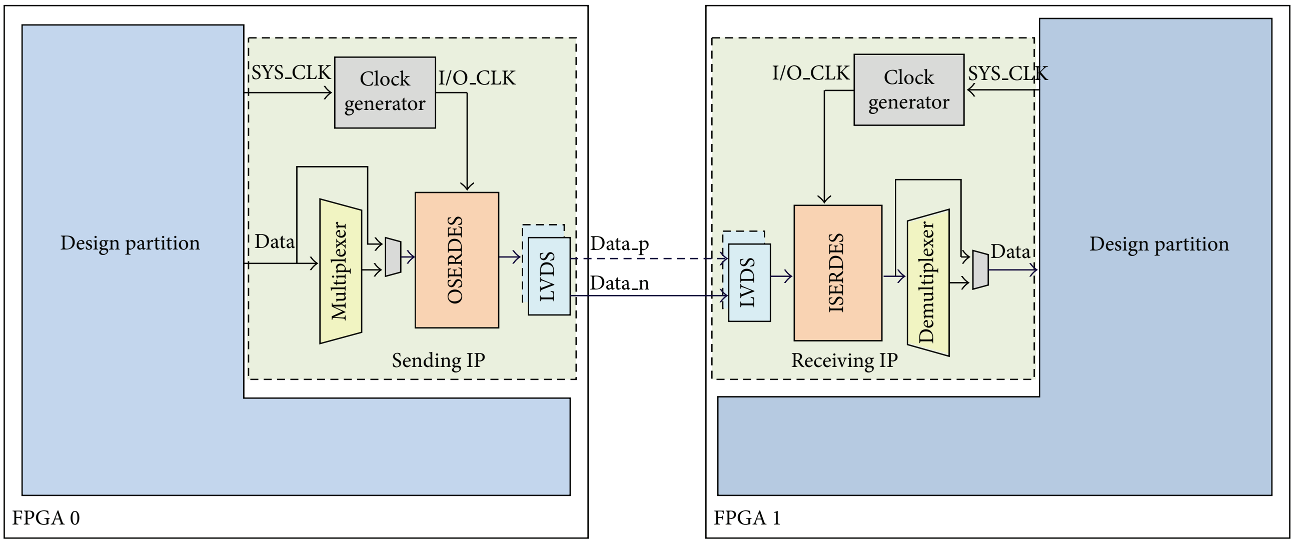


Figure 11: Multiplexing IP architecture.

Table 1: Comparison between routing results of WASGA and CERTIFY partitioning tools.

Benchmarks	WASGA				CERTIFY			
	Cut signals	NB_FPGA	R_hop	Mux_ratio	Cut signals	NB_FPGA	R_hop	Mux_ratio
CPU20_occ10	1545	6	0	3	3316	6	0	10
CPU20_occ20	1002	4	0	3	1634	4	0	4
CPU30_occ20	1710	4	0	3	3076	4	0	6
CPU30_occ30	1487	4	0	4	2521	3	0	7
CPU50_occ30	2819	4	0	5	5279	4	0	11
CPU50_occ50	2202	4	0	6	4019	3	0	9
CPU125_occ50	7809	6	1	11	NR	NR	NR	NR
CPU125_occ65	7644	5	0	12	NR	NR	NR	NR

of managed elements. The targeted multi-FPGA prototyping board that we use for the experiments is a DNV6F6PCIe from the Dini group [17]. As shown in Figure 12, this board contains 6 FPGAs Virtex-6 LX550T using all the same package FF1759, meaning that they have the same number of total user I/Os. The inter-FPGA clock frequency is set to 500 MHz. Applying this frequency on the multiplexing IPs (ISERDES/OSERDES with LVDS), the inter-FPGA communication data rate on this board is 1 Gbps using double data rate (DDR).

To map the designs into this board, we use the WASGA partitioning flow provided by Flexras Technologies [19]. WASGA partitions the designs and outputs the list of inter-FPGA signals that should be routed. After routing these signals, using the routing methodology detailed in this paper, WASGA generates a netlist for each FPGA which contains the multiplexing IP to ensure the transmission of the multiplexed signals. The resulting netlists are entered into the FPGA flow to execute the place and route and the bitstream generation individually for each FPGA.

Firstly, we set the constraints listed in Section 3.2 to the WASGA partitioning tool. Table 1 presents the routing results obtained by WASGA flow and CERTIFY partitioning tool [23]. The number of cut signals obtained by the WASGA partitioner is considerably less than the number of the signals obtained by CERTIFY. WASGA aims to optimize the number of combinational hops. Therefore, for all the tested designs, the mux_ratio results are improved compared to those obtained by CERTIFY. Table 1 shows results related to the number of routing hops used in each benchmark. The number of routing hops is the number of FPGAs crossed by a signal from the source until reaching its destination. Results presented in Table 1 reflect the importance of partitioning on the system frequency. We should notice that for the 2 last benchmarks, the designs are not routable (NR) with the CERTIFY tool. In fact, since the number of cut signals is becoming larger, the mux_ratio is more and more important. CERTIFY provides multiplexing IP with a maximum width equal to 32 [24]. Thus, all the designs which need a multiplexing ratio ≥ 32 are not routable.

On the other hand, we tried to select the best shape of the routing signals. Table 2 shows the results for each routing scenario described in Section 5. These scenarios are defined depending on the signal shape and the routing graph. Four scenarios are selected to test the performance of the iterative routing algorithm on the multi-FPGA prototyping platform.

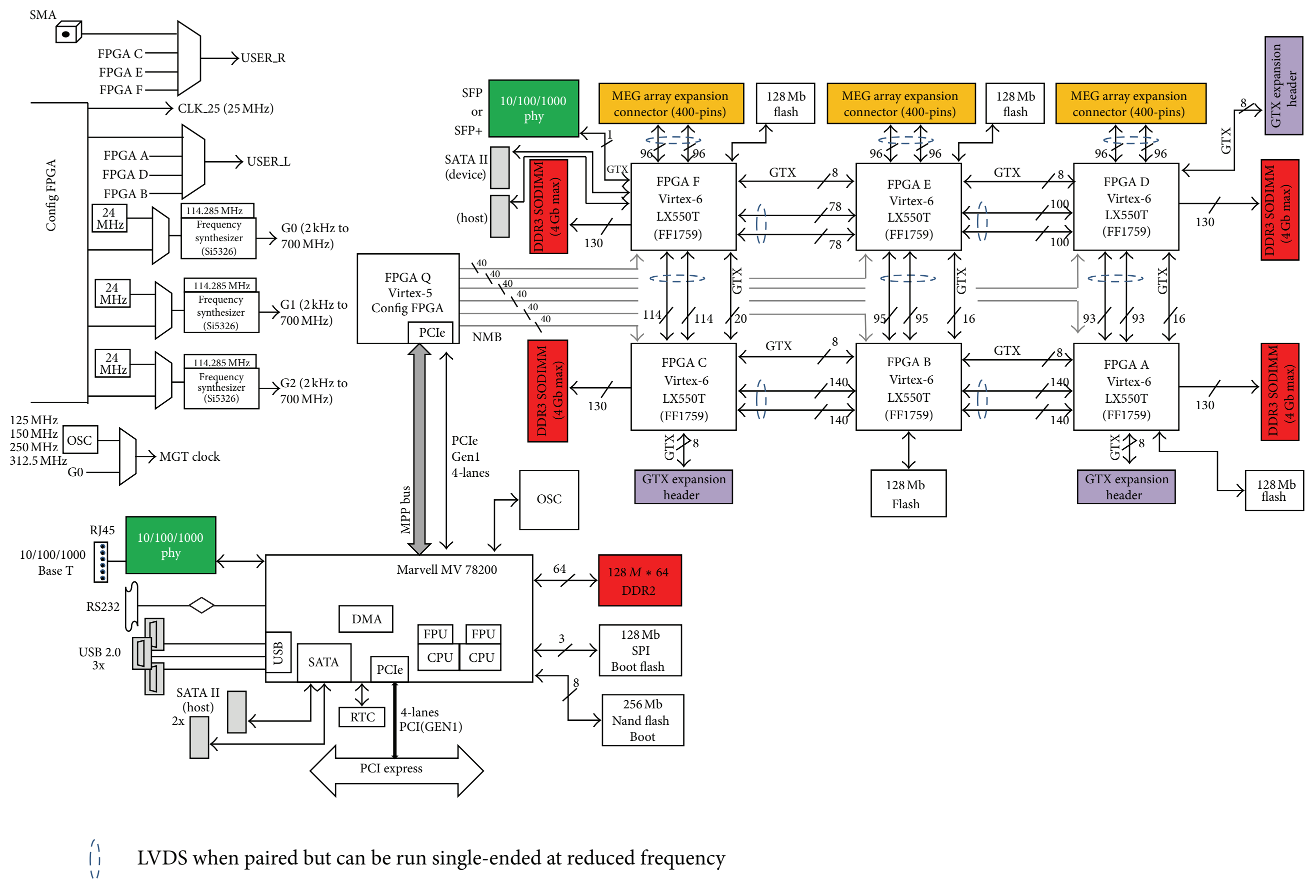

LVDS when paired but can be run single-ended at reduced frequency
GTX: Packet I/O transceivers 6.5 Gb/s per channel bidirectional (13 Gb/s max)
FPGA F Virtex-6 LX550T (FF1759)
FPGA E Virtex-6 LX550T (FF1759)
FPGA D Virtex-6 LX550T (FF1759)
FPGA C Virtex-6 LX550T (FF1759)
FPGA B Virtex-6 LX550T (FF1759)
FPGA A Virtex-6 LX550T (FF1759)
FPGA Q Virtex-5 Config FPGA
Marvell MV 78200
MEG array expansion connector (400-pins)
DDR3 SODIMM (4 Gb max)
GTX expansion header
128 Mb flash
PCI express
MPP bus
PCIe Gen1 4-lanes
Config FPGA
MGT clock
USER_R
USER_L
CLK_25 (25 MHz)

Figure 12: Prototyping board based on six Virtex-VI from Dini group [17].

Table 2: Comparison of routing strategies effects on prototyping system performance.

Benchmarks	Scenario 1			Scenario 2			Scenario 3			Scenario 4		
	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

In these experiments, we use benchmarks where 70% of signals are multiterminal ones.

(i) In scenario 1, multiterminal signals are routed on a unidirectional routing graph.

(ii) In scenario 2, two-terminal signals are routed into a unidirectional routing graph where the nodes capacity can be greater or equal to 1.

(iii) In scenario 3, multiterminal signals are grouped into GSignals. These GSignals are routed into a bidirectional routing graph where all node capacities are set to 1.

(iv) Finally, in the fourth scenario, the two-terminal branches are combined into groups and routed into a bidirectional routing graph.

Results show that routing on a bidirectional graph gives much better results since the router has more flexibility 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 large number of routing hops penalizes the system frequency.

Table 3: 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%

Since we have demonstrated that scenario 4 gives usually the best results, we apply Pathfinder and the obstacle avoidance routing algorithms (constructive algorithms) to route inter-FPGA signals, all with one source and one destination (branch), and grouped into GSignals. Table 3 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 negotiation 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 trade-off between maximum multiplexing ratio and routing hops.

8. Conclusion

Prototyping is no longer optional due to the cost of chips and difficulty to simulate huge designs. To validate designs more efficiently, the highest frequency should be reached. The system frequency depends on the way the inter-FPGA signals are routed. In this paper, we presented our approach to route these inter-FPGA signals. We set a number of constraints to the partitioning tool in order to get the best partitioning solution which leads to the optimal routing. We extend the Pathfinder routing algorithm to deal with the inter-FPGA signals. In order to select the best signal shape, we tested the performance of this iterative routing algorithm on four scenarios.

The best scenario in terms of system frequency consists in grouping signals 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%.

Acknowledgment

This research paper is made possible through the help and support of the Feder European Grant.

References

[1] C. Huang, Y. Yin, and C. Hsu, "SoC HW/SW verification and validation," in *Proceedings of the 16th Asia and South Pacific Design Automation Conference (ASP-DAC '11)*, pp. 297–300, January 2011.

[2] M. Santarini, "ASIC prototyping: make versus buy," *EDN*, vol. 50, no. 24, pp. 30–40, 2005.

[3] D. Amos, A. Lesea, and R. Richter, *FPGA-Based Prototyping Methodology Manual*, Synopsys, 2011.

[4] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," in *Proceedings of the 14th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 21–30, February 2006.

[5] H. Krupnova, "Mapping multi-million gate SoCs on FPGAs: industrial methodology and experience," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 2, pp. 1236–1241, February 2004.

[6] S. Asaad, R. Bellofatto, B. Brezzo et al., "A cycle-accurate, cycle-reproducible multi-FPGA system for accelerating multicore processor simulation," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '12)*, pp. 153–162, February 2012.

[7] J. Babb, R. Tessier, M. Dahl, S. Z. Hanono, D. M. Hoki, and A. Agarwal, "Logic emulation with virtual wires," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 6, pp. 609–626, 1997.

[8] L. McMurchie and C. Ebeling, "PathFinder: a negotiation-based performance-driven router for FPGAs," in *Proceedings of the International Workshop on Field Programmable Gate Array*, pp. 111–117, February 1995.

[9] A. Ejnioui and N. Ranganathan, "Multiterminal net routing for partial crossbar-based multi-FPGA systems," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 11, no. 1, pp. 71–78, 2003.

[10] X. Song, W. N. N. Hung, A. Mishchenko, M. Chrzanowska-Jeske, A. Kennings, and A. Coppola, "Board-level multiterminal net assignment for the partial cross-bar architecture," *IEEE Transactions on Very Large Scale Integration ystems*, vol. 11, no. 3, pp. 511–513, 2003.

[11] W. Mak and D. F. Wong, "Board-level multiterminal net routing for FPGA-based logic emulation," *ACM Transactions on Design Automation of Electronic Systems*, vol. 2, no. 2, pp. 151–157, 1997.

[12] W. Mak and D. F. Wong, "On optimal board-level routing for FPGA-based logic emulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 3, pp. 282–289, 1997.

[13] J. Babb, R. Tessier, and A. Agarwal, "Virtual wires: overcoming pin limitations in FPGA-based logic emulators," in *Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines (FCCM '93)*, pp. 142–151, April 1993.

[14] R. Tessier, J. Babb, M. Dahl et al., "The virtual wires emulation system: a gate-efficient ASIC prototyping environement," in *Proceedings of the International Workshop on Field-Programmable Gate Array*, ACM, Berkeley, Calif, USA, February 1994.

[15] M. Inagi, Y. Takashima, Y. Nakamura, and A. Takahashi, "Optimal time-multiplexing in inter-FPGA connections for accelerating multi-FPGA prototyping systems," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences A*, vol. E91, no. 12, pp. 3539–3547, 2008.

[16] M. Inagi, Y. Takashima, and Y. Nakamura, "Globally optimal time-multiplexing in inter-FPGA connections for accelerating multi-FPGA systems," in *Proceedings of the 19th International Conference on Field Programmable Logic and Applications (FLP '09)*, pp. 212–217, September 2009.

[17] [Online], http://www.dinigroup.com/new/products.php.

[18] Synopsys FPGA Synthesis User Guide, 2011.

[19] [online], http://www.flexras.com/.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, London, UK, 2001.

[21] M. Turki, Z. Marrakchi, H. Mehrez, and M. Abid, "Iterative routing algorithm of Inter-FPGA signals for Multi-FPGA prototyping platform," in *Proceedings of the 9th international conference on Reconfigurable Computing (ARC '13)*, Los Angeles, Calif, USA, March 2013.

[22] M. Turki, Z. Marrakchi, H. Mehrez, and M. Abid, "Towards synthetic benchmarks generator for CAD tool evaluation," in *Proceedings of the 8th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME '12)*, 2012.

[23] [Online], http://www.synopsys.com/Systems/FPGABasedPrototyping/Pages/Certify.aspx.

[24] Certify Partition Driven Synthesis, User Guide, March 2011, p. 153.