Inter-FPGA interconnect topologies exploration for multi-FPGA systems

Umer Farooq, Habib Mehrez, Muhammad Khurram Bhatti

Des Autom Embed Syst (2018) 22:117–140
https://doi.org/10.1007/s10617-018-9207-2

Inter-FPGA interconnect topologies exploration for multi-FPGA systems

Umer Farooq[1] · **Habib Mehrez**[2] · **Muhammad Khurram Bhatti**[3]

Received: 14 August 2017 / Accepted: 20 April 2018 / Published online: 2 May 2018

Abstract Prototyping using multi-FPGA systems offers significant advantages over simulation and emulation based pre-silicon verification techniques. Multi-FPGA prototyping follows a complex design flow where the quality of associated tools and the architecture of interconnect topology play a very important role in the performance of final prototyped design. A well designed interconnect topology may remain underutilized because of a poor routing tool and vice versa. This makes the selection of a good routing tool and the exploration of interconnect topologies extremely important for the quality of final design. In this work, we present a detailed comparison between six inter-FPGA interconnect topologies. We present a generic routing tool and for each topology, ten large, complex benchmarks are prototyped on four FPGA boards using this tool. Experimentation reveals that fully customized interconnect topology using a hybrid combination of direct two and multi point tracks gives the best frequency results for all the FPGA boards. On average, this topology gives 26.2, 28.5, 9.5, 32.1 and 12.4% better frequency results as compared to five other interconnect topologies. We also perform routing time comparison and the topology using generic hybrid combination of direct two and multi point tracks gives the best results. On average, this topology produces 1.8×, 2×, 2×, 9.2×, and 4.4× better results as compared to five other topologies under consideration. Frequency–time tradeoff analysis along with flexibility and setup time of different topologies is also performed. It reveals that a partially customized topology with hybrid combination of direct two and multi point tracks gives the best frequency–time tradeoff for smaller FPGA boards while a partially customized topology

✉ Umer Farooq
ufarooq@du.edu.om

Habib Mehrez
habib.mehrez@lip6.fr

Muhammad Khurram Bhatti
khurram.bhatti@itu.edu.pk

1 Electrical and Computer Engineering Department, Dhofar University, Salalah, Oman

2 CNRS, LiP6, Sorbonne Universités, UPMC University Paris 06, Paris, France

3 Electrical Engineering Department, Embedded Computing Lab, ITU, Lahore 54000, Pakistan

with switch-based and multi point connections gives the best results for larger FPGA boards with reasonable flexibility and moderate setup time.

Keywords Multi-FPGA prototyping · Interconnect topologies · Routing · Exploration

1 Introduction

The advancement of processing technology and improved design tools have tremendously increased the computation capability of modern digital systems. This, however, has come at the cost of complex, lengthy and expensive design process of digital systems [1,2]. Today, it takes about two years to roll out the first prototype of a complete digital system while requiring hundreds of thousands of dollars in capital investment [2]. Moreover, the shrinking time to market window, the faster bring-up time pressure, and the weak reliability index of modern processing technologies further exacerbates the situation. In this scenario, pre-silicon verification becomes an important step in the design process of a digital system as it can accelerate the design process of first prototype and can also save stakeholders from monetary loss and eventual embarrassment of a faulty product launch [3,4]. Commonly used pre-silicon verification techniques are simulation, emulation, and Field Programmable Gate Array (FPGA) -based prototyping [5]. Each of these techniques has its advantages and disadvantages. For example, simulation based verification options [6–8] offer quick setup time with complete system visibility. However, their execution speed is quite low and run time for complex systems can take several days or even weeks. Emulation based platforms [9–11] offer better execution speed as compared to simulators with complete system visibility, huge logic capacity and debugging capability. They are, however, prohibitively expensive and require large setup time. Compared to simulation and emulation, FPGA-based prototyping gives the best execution speed while offering similar logic capacity and set-up time as that of emulators. A unique advantage of FPGA-based prototyping is its ability to execute the design at cycle-accurate, bit-accurate level with real world interfacing experience. A comparison of the characteristics of three verification techniques is shown in Fig. 1. It can be seen from this figure that advantages like frugality, high performance and real world testing experience make FPGA-based prototyping most favored among the three verification techniques. But, because of the huge logic requirements of modern digital systems and large area gap between FPGAs and Application Specific Integrated Circuits (ASIC) [12], single FPGA boards can not be used for the prototyping of complex systems; thus paving the way for multi-FPGA boards.

Prototyping using multi-FPGA systems is a tough task as its back-end flow follows a complex process. The back-end flow starts with the synthesis of Register Transfer Level (RTL) description of the design under consideration. After synthesis, the design is partitioned over multiple-FPGAs. The objective of this step is to divide the design in such a way that each part fits in the logic capacity of target FPGA while the communication between different partitions is kept as small as possible. Next, inter-FPGA routing of the signals (also termed as cut-nets) that traverse different partitions is performed in Time Division Multiplexed (TDM) manner [13]. Finally, the intra-FPGA placement and routing is performed by vendor specific tools and bitstreams of the partitioned design are loaded onto respective FPGAs of multi-FPGA board. The number of FPGAs on the multi-FPGA board normally depends upon the complexity of design under consideration. Their number may vary from a few FPGAs on a single board [14] to several dozen FPGAs on multiple FPGA boards [15].

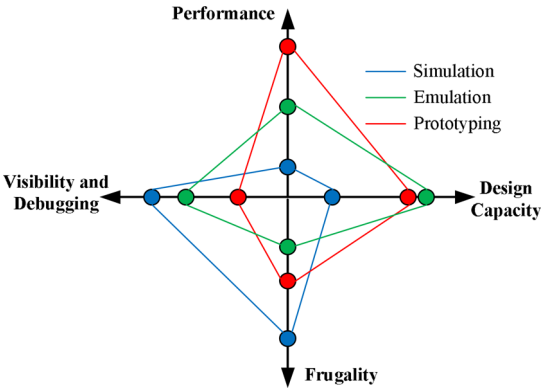

Fig. 1 Comparison of different verification techniques

In multi-FPGA prototyping flow, the quality of inter-FPGA routing interconnect and the capability of corresponding inter-FPGA routing tool to exploit the characteristics of routing interconnect are very vital. This is because of the fact that a well designed routing interconnect might remain underutilized due to a poorly conceived inter-FPGA routing tool and vice versa. In multi-FPGA prototyping flow, when a design is partitioned, the number of cut-nets traversing different partitions outnumber the available physical tracks between different partitions (i.e. FPGAs). So, the cut-nets having same source and destination are multiplexed together in a time division manner, sent over the available physical tracks, and then demultiplexed at the receiving end. The aforementioned process is performed on an interconnect topology using an inter-FPGA routing tool. Normally, the maximum number of cut-nets traversing through a single physical track are termed as the multiplexing ratio (also termed as mux ratio in this work) of the design and it directly affects the execution speed of final prototyped design. Sometimes, if the direct path from the source to the destination FPGA is not available, then an intermediate FPGA might also be used as a hop. Addition of hops in the routing path further deteriorates the execution speed of the final prototyped design. The problem of multiplexing ratio and hops becomes particularly relevant in the context of modern day FPGAs where logic capacity has enormously increased as compared to their previous generations while available I/Os have either remained same or decreased. Table 1 gives an overview of evolution of logic capacity and number of I/Os in Xilinx's Virtex and Altera's Stratix FPGA families over the past few years. Different studies have been carried out in the past on interconnect routing topologies and inter-FPGA routing approaches. These studies have mainly aimed at improving the execution speed by reducing the multiplexing ratio and the number of intermediate hops by using fixed routing topologies and approaches. However, these studies are too specific in their nature (further details in Sect. 2) and a detailed exploration, comparison between different routing topologies and an in-depth analysis of the results has not been performed before.

In this work, we explore six different inter-FPGA interconnect topologies through our indigenously developed, generic inter-FPGA routing tool. These topologies are broadly categorized into off-the-shelf and custom interconnect topologies. Off-the-shelf interconnect topologies are further divided into direct and switch-based interconnect while custom topologies are divided into partial- and full- custom categories. Architectural details of these topologies and relevant routing approaches are presented in the following sections of the paper. For exploration of topologies, we use four different FPGA boards and a suite of large benchmarks to perform detailed comparison between different routing topologies. These

Table 1 FPGA logic capacity to I/O ratio for different FPGAs

FPGA family name	FPGA device name	No. of I/Os	Gates per I/O
Virtex 4	XC4VLX100	960	3000
Virtex 5	XC5VLX220	1200	2900
Virtex 6	XC6VLX550T	1200	8000
Virtex 7	XC7V2000T	1200	20,000
Stratix 2	EP2S180	1170	2000
Stratix 3	EP3SL340	1104	4000
Stratix 4	EP4S100G5	1120	9500
Stratix 5	5SEEB	840	16,000

benchmarks mimic real life applications and most of them are generated using a tool developed locally at our lab. Further details about the benchmarks and their generation mechanism are given in the next section of the paper. To the best of our knowledge, this kind of detailed exploration and comparison of inter-FPGA routing has not been performed before. The main contributions of this work are summarized below.

- Generation of complex, practical benchmarks through benchmark generation tool.
- A generic exploration flow using which six different inter-FPGA interconnect topologies are explored through locally developed, generic inter-FPGA routing tool.
- Extensive experimentation using the benchmarks, routing tool and profound analysis of the results obtained for different interconnect topologies.

The rest of the paper is organized as follows. Section 2 presents a detailed discussion on the state-of-the-art work related to this paper and also elaborates the contributions of this work. Section 3 presents multi-FPGA prototyping flow that is used in this work. This section also discusses in detail the inter-FPGA routing tool used in this work. Section 4 discusses proposed exploration topologies. Section 5 presents a comprehensive analysis of experimental results obtained through exploration and this paper concludes in Sect. 6 with discussion on the future work.

2 Related work

As discussed in Sect. 1, in this work, we explore and compare six different interconnect topologies through our generic inter-FPGA routing tool. For multi-FPGA interconnect topology and routing tool testing, large, complex and real life benchmarks are a major requirement. Previous studies exist where authors have used different benchmark sets for their experimentation. For example MCNC [16] benchmarks have long been used by the research community in various research studies. Although complex in nature, these benchmarks are quite small and are unable to challenge the capacity handling capability of modern day tools. Similarly, authors in [17] use complex heterogeneous benchmarks but they are small in size. The matter of benchmark size is addressed in [18] where authors present a synthetic benchmark generator that can generate benchmarks of quite large sizes. However, the generated benchmarks have redundant components with repetitive connections and they possess little or no similarity to real world applications. In this work, we use the Design Space eXploration (DSX) [19] tool developed at our lab to generate different multi-clustered Multi Processor System on

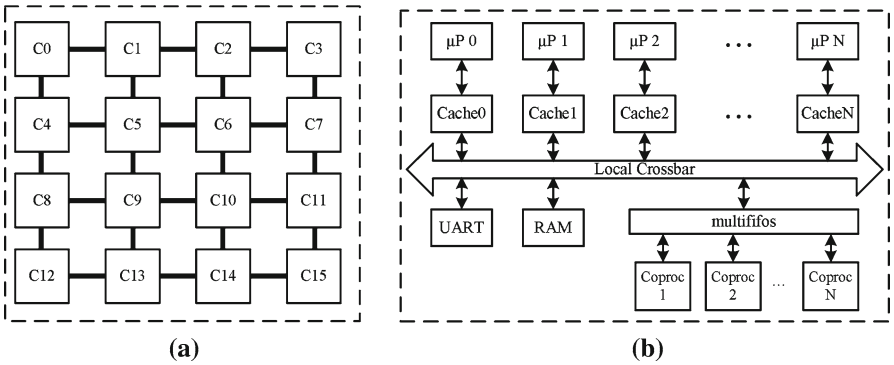

Fig. 2 **a** Multi-clustered MPSoC architecture overview; **b** internal architecture of a single cluster of MPSoc

Chip (MPSoC) architecture benchmarks. These multi-clustered benchmarks are large in size and they mimic real life applications. Different clusters in the multi-clustered MPSoC are arranged in a two dimensional grid and they communicate with each other through DSPIN NoC [20]. Furthermore, each cluster in a multi-clustered MPSoC contains multiple microprocessors which communicate with other components of the cluster through local crossbar. Figure 2a gives an abstract overview of multi-clustered MPSoC architecture while Fig. 2b gives internal details of a single cluster. It can be seen from these figures that multi-clustered MPSoC can have a large mesh of clusters and inside each cluster we can have different components like processors, RAM, UART etc. In this work, we generate multi-clustered MPSoCs with varying sizes and due to their large logic requirement and complex structure, they make a perfect case for interconnect topology exploration and inter-FPGA routing tool testing. Further details about the benchmark generation process are given in Sect. 3.1 of the paper.

Once the benchmark is generated, it is synthesized, partitioned [21] and routed using inter-FPGA routing tool. Different approaches have been presented in the past for inter-FPGA routing. For example, an obstacle avoidance inter-FPGA routing approach based on integer linear programming is presented in [22]. This technique generates good routing results in a short time. But because of its obstacle avoidance approach, it is prone to fall in the local minima and report a problem even if there exists a feasible solution. This problem is tackled in [23] where authors propose a negotiation based, congestion-driven inter-FPGA routing technique. This technique, however, takes more time as compared to [22] to find a feasible solution. When a benchmark is partitioned into multiple parts, the resulting cut-nets can be either two terminal (having single source and single destination) or they can be multi terminal (having single source and multiple destinations). Similarly, a multi-FPGA board can have only either two point interconnect or it can have a mixture of two point and multi point connections [24]. The routing technique presented by [23] considers multi-FPGA boards with two point tracks only. To route multi terminal nets on two point tracks, authors in [23] first decompose them into two terminal nets and then perform the routing. This routing deficiency is addressed by [25] where two terminal, multi terminal nets are routed separately on two point and multi point tracks. The technique in [25] has reported to produce better performance results as compared to [23].

All the work cited above considers fixed two point only or mixed two, multi point interconnect topologies where no exploration is performed to investigate the effect of FPGA board architecture on the execution speed of final prototyped design. In this regard, authors

in [26] present detailed exploration of the effect of multi-FPGA board size on the frequency of target design. They also perform an exploration on the effect of the percentage of two and multi point tracks on the frequency of target design. However, this study is also based on fixed inter-FPGA routing interconnect only. Authors in [27] propose partial cross bar switch-based configurable routing interconnect for multi-FPGA systems and authors in [28] explore mesh-based, configurable inter-FPGA interconnect topology. But they evaluate their approaches empirically and do not perform any experimentation. Similarly, authors in [29] propose to use a hybrid partial cross bar as inter-FPGA routing interconnect for multi-FPGA systems. Their experimental setup, however, uses quite small benchmarks [16] for testing and no comparison to other interconnect topologies is performed.

Contrary to the work cited above, in this work, we propose to explore six different interconnect topologies. Through this exploration, we have unearthed different interconnect topologies' behavior for different benchmarks on varying FPGA board sizes. These topologies are explored using our indigenous inter-FPGA routing tool and large benchmarks are used for experimentation. To the best of our knowledge, no prior study exists that performs such an extensive exploration and comparison between different inter-FPGA routing interconnect topologies for varying FPGA board while using such a large set of benchmarks. Among the six techniques under consideration, four use either direct two point only, or a combination of direct two and multi point connections. These four techniques are further divided into two off-the-shelf, one partially customized, and one fully customized technique. It is important to mention here that this work is an extension of [26] and for partially customized technique, we leverage from exploration results of [26]. However, the results of remaining three techniques are entirely new and are being presented for the first time in this work. As for the remaining two techniques of the six techniques under consideration, we use partial cross bar based configurable switch as the routing interconnect for multi-FPGA systems. We use two different variants of switch based topology. In the first variant all cut-nets (be it two terminal or multi terminal) are routed through a disjointed, partial cross bar reconfigurable switch. In the second variant, two terminal cut-nets are routed through switch box whereas multi terminal cut-nets are routed through direct connections. The main objective to use the switch based routing is to give the flexibility to the routing interconnect because switch-based reconfigurable communication networks have reported to produce good routing results while requiring small area [30]. Six topologies described above are explored using a back-end prototyping flow. In the next section, we discuss different steps of the flow and then also present the architectural details of six topologies in Sect. 4 of the paper.

3 Exploration flow

For the exploration of different inter-FPGA interconnect topologies, we have developed a generic exploration flow. The main objective of this flow is to optimally prototype a design on multi-FPGA board. For this purpose, we have integrated some industrial and indigenously developed tools. When combined together, these tools give complete multi-FPGA prototyping experience. An overview of the flow used in this work is shown in Fig. 3. It can be seen from this figure that the flow starts with the generation of benchmarks. These benchmarks are then synthesized and partitioned. After partitioning, inter-FPGA routing is performed. It is at this step that we mainly perform exploration of different interconnect topologies. For a given multi-FPGA board, the objective of inter-FPGA routing is to find the optimal multiplexing ratio (i.e. mux ratio) for a particular benchmark as it directly impacts the frequency of

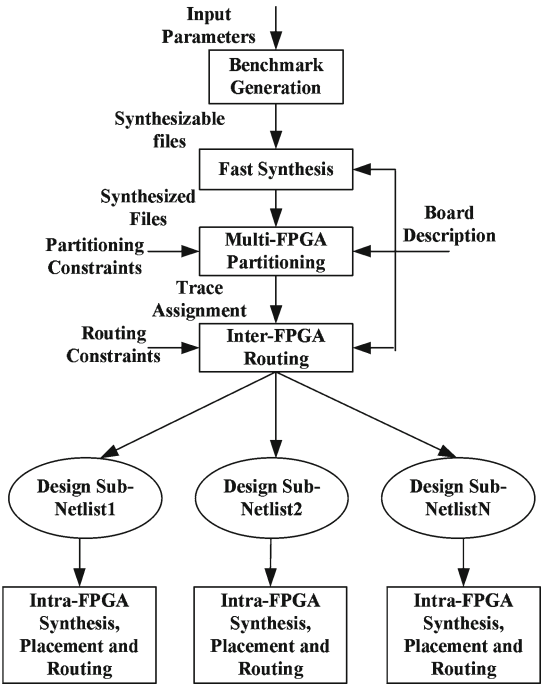

Fig. 3 An overview of exploration flow

final prototyped design. Once inter-FPGA routing is done, the frequency of the design is determined and flow is terminated after intra-FPGA placement and routing. Further discussion about different steps of the design flow are given in the following sub-sections.

3.1 Benchmark generation and synthesis

As explained in Sect. 2, in this work, we generate large MPSoC benchmarks through locally developed DSX tool. This tool takes three parameters as input. First parameter contains the description of MPSoC architecture and the instantiations of all the components contained in the architecture. For example, this parameter contains number of cluster in the architecture, number of processors inside each cluster, size and type of other interface components like UART, RAM etc inside each cluster. Second parameter then gives details of all the components of the architecture and also defines the interface between them. Third parameter finally defines the software application graph with variable number of tasks running on different processors of MPSoC architecture. Once these parameters are given to the DSX tool, it automatically generates MPSoC architecture's synthesizable files. These files are then given to Certify [21] that performs their fast synthesis. This tool logically optimizes the design and maps it to the library of target FPGA architecture. Information about the target FPGA architecture comes from the board description file. After synthesis, the design is partitioned by the same tool.

3.2 Multi-FPGA partitioning

To partition the design under consideration, we use an industrial tool (i.e. Certify) that takes board description, partitioning constraints, and synthesized files as input (see Fig. 3). The board description file gives information about the family of target FPGA architecture as well

as the number of routing tracks available between different FPGAs on board. The constraints file specifies the size of each partition, partitioning objectives like the balance factor, and the optimization objectives like minimum cut-nets etc. The information provided by the board description file and the constraints file is used by partitioning tool to partition the design into the required number of parts. While partitioning, the principle objective of the tool is to fit each partition in the specified logic capacity while minimizing the total number of cut-nets. The output of the tool is the partitioned design and the trace assignment file. This file contains all the information about the two point and multi point cut-nets of the partitioned design. This file is then given to routing tool to perform inter-FPGA routing.

3.3 Inter-FPGA routing

Inter-FPGA routing plays a very crucial role in the execution speed of final prototyped design. Here, we give details about a generic inter-FPGA routing tool developed indigenously. Using this tool, we explore six interconnect topologies described in Sect. 4 for four different FPGA boards. In Fig. 3, the inter-FPGA routing is presented as a block. However, its detailed flow is presented in Fig. 4. It can be seen from this figure that this tool takes three files as input. First one is the board description file that gives information about available board resources, the type of interconnect topology and the number of FPGAs on board. The second is the routing constraint file that gives constraints like routing type (timing-driven or routability-driven), optimization approach (binary search or sequential search). The third is a trace assignment file that contains information about all the cut-nets of the partitioned design. Inter-FPGA routing tool takes these files and performs different steps to terminate with frequency estimation of design under consideration for a specific topology. The objective of the routing tool is to maximize the system frequency by minimizing the mux ratio and the number of hops. Interconnect topology exploration is mainly performed at this step where all the partitioned benchmarks are routed for each topology and this process is repeated for different FPGA boards under consideration. Further details about the different steps of inter-FPGA routing are given next.

3.3.1 Graph generation

It is shown in Fig. 4 that first of all routing tool generates the routing graph where information for the graph is taken from board description. In this graph, board resources are presented as an abstracted graph $G(V, E)$ where vertices V represent the I/O resources of the FPGAs on board while edges E represent the potential connections between the I/Os. Figure 5a shows sample representation of a four FPGA board having various two point and multi point tracks. Routing graph representation of the sample board is given in Fig. 5b. Routing graph of the physical resources generated in this step is later used by Pathfinder [31] routing algorithm to perform routing of cut-nets on the tracks of FPGA board.

3.3.2 MUX ratio computation and cut net grouping

Once the routing graph is generated, mux ratio is next computed as the ratio of maximum number of cut-nets between two FPGAs and the number of physical tracks between corresponding FPGAs. Based on the computed mux ratio value, the cut-nets having the same source and destination are combined together so that they may be later routed on a single FPGA board track using TDM.

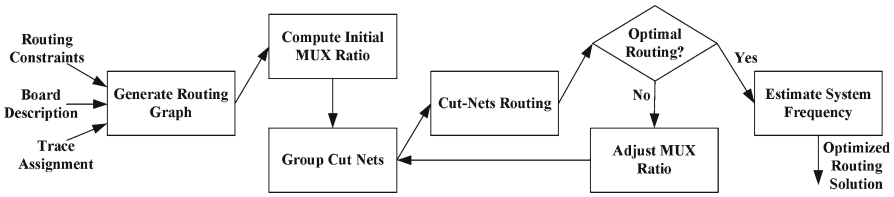

Fig. 4 An overview of inter-FPGA routing flow

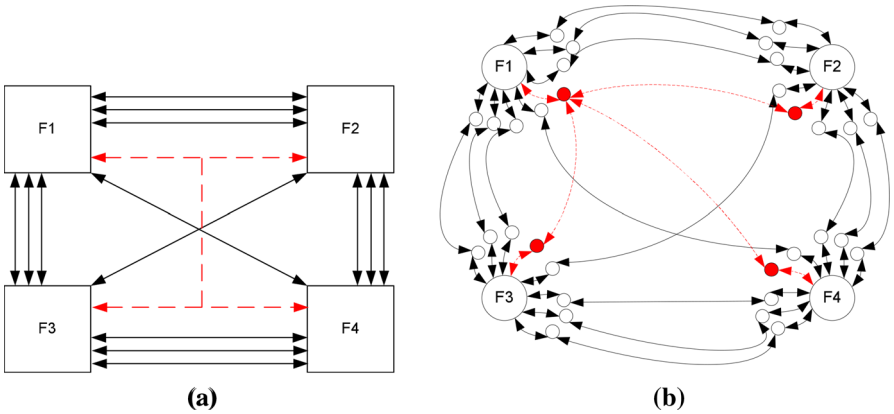

Fig. 5 **a** Physical resource description of a sample four FPGA board with two point and multi point tracks; **b** routing graph representation of sample four FPGA board

3.3.3 Cut-net routing

After grouping, cut-nets are routed on the FPGA tracks using Pathfinder [31] routing algorithm. Pathfinder is a negotiation based, congestion-driven routing algorithm that routes groups of cut-nets, one at a time, using Dijkstra's shortest path algorithm [32]. In this work, we propose to use the timing-driven routing approach for Pathfinder algorithm as it gives same or better routing results as compared to the routability-driven approach while requiring smaller routing time [26]. The congestion resolution cost function of timing-driven routing approach is given in Eq. 1. In this equation, $b(n)$ is the base cost of congested node n and normally its value is set to be 1 at the start of routing process. In Eq. 1, $h(n)$ is the historical cost of congested node and its record is maintained so that a previously congested node is not selected again for future iterations, $p(n)$ is the present cost factor that gives congestion value of a node in the current iteration and $Crit(i, j)$ is the criticality of connection from source FPGA i to destination FPGA j. Criticality of a connection is further calculated using Eq. 2 where $slack(i, j)$ is the amount of delay that could be added to node before it affects the critical path delay and $Dmax$ is the circuit critical path delay. The cost function of Eq. 1 is used to find a conflict free solution at a particular mux ratio while using the minimum number of intermediate hops which is ensured through Dijkstra's shortest path algorithm.

$$cost(n) = Crit(i, j)xdelay(n) + [1 - Crit(i, j)]x[b(n) + h(n)]xp(n) \quad (1)$$

$$Crit(i, j) = 1 - \frac{slack(i, j)}{Dmax} \quad (2)$$

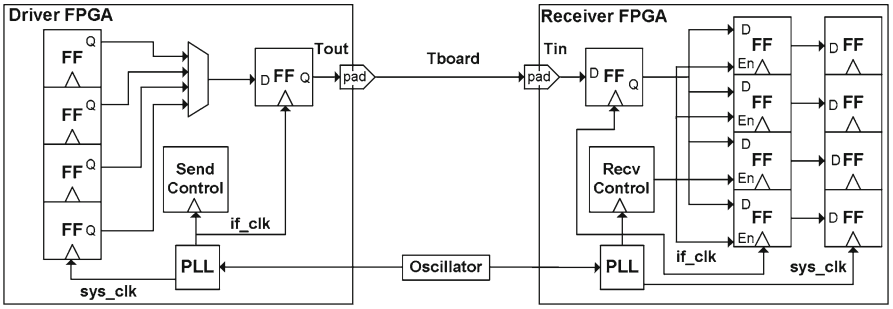

Fig. 6 Frequency estimation in multi-FPGA board

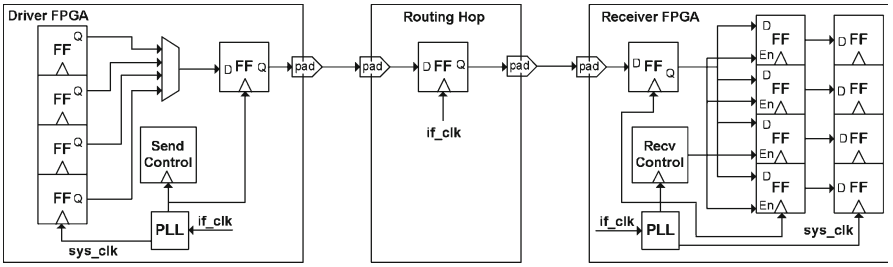

Fig. 7 Frequency estimation in multi-FPGA board with routing hop

3.3.4 Mux ratio optimization and frequency estimation

It can be seen from Fig. 4 that after the cut-nets are routed, the optimality of the routing solution is checked in terms of mux ratio. In this work, we optimize the mux ratio value through binary search algorithm. Compared to the sequential search, the binary search algorithm gives same result with fewer routing rounds; hence optimizing the overall routing time [26]. Once the mux ratio is optimized, the frequency estimation of the design under consideration is performed using Eq. 3.

$$sys\_freq = \frac{if\_freq}{mux\_ratio + hops}\text{MHz} \tag{3}$$

The concept of if_freq can be understood from Fig. 6 where $if\_freq = 1/T_{\text{crit}}$ and $T_{\text{crit}} = T_{\text{out}} + T_{\text{board}} + T_{\text{in}} + T_{\text{tolerance}}$. Although our inter-FPGA routing tool tries to minimize the number of hops through its shortest path algorithm, intermediate FPGAs may still become necessary if there are no direct tracks available between a driver FPGA and the receiver FPGA at a particular mux ratio. In such a case, intermediate FPGAs become routing hops. Graphical representation of intermediate FPGA acting as routing hop is shown in Fig. 7. In this work, for direct connections, we assume an $if\_freq = 125$ MHz while for switch-based interconnect topology internal delay of reconfigurable switch is added that increases the critical path delay and reduces the if_freq to 111 MHz.

3.4 Intra-FPGA place and route

It can be seen from Fig. 3 that after the optimized routing, partitions of the design under consideration are combined to generate design sub-netlists.These design sub-netlists are synthesized, placed and routed on the target FPGA architecture with the help of vendor

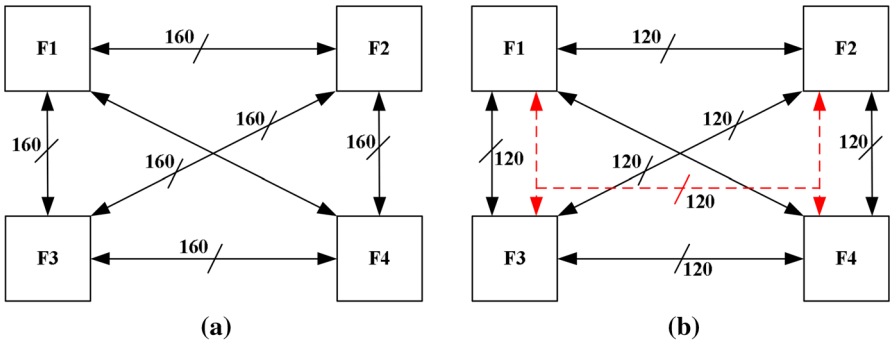

Fig. 8 **a** Two point generic multi-FPGA board (topology 1); **b** two point and multi point generic multi-FPGA board (topology 2)

specific tools. Most commonly used tools are Vivado [33] for Xilinx FPGAs and Quartus Prime [34] for Altera FPGAs. In this work, Quartus Prime tool by Altera is used that places and routes each partition on the FPGA and generates the corresponding bitstream which is finally loaded onto the FPGA for debugging.

4 Interconnect topologies

In this work, we explore six different interconnect topologies using the exploration flow described in Sect. 3. The explored topologies can either be categorized as direct, switch-based topologies or they can also be classified as off-the-shelf and custom interconnect topologies. Further details of these topologies are as follows.

4.1 Generic, direct, two point only interconnect topology (topology 1)

The first topology that we explore in this work is a generic interconnect topology that supports only two point connections between different FPGAs on board. This is an off-the-shelf interconnect topology where connections between different FPGAs on the board are pre-defined and they are distributed in a balanced way. Figure 8a shows an example of this topology. This kind of topology is mostly used in commercially available FPGA boards [24] as it is easy to fabricate but it produces poor frequency results. As explained in Sect. 2, a partitioned design can have both two terminal and multi terminal cut-nets. In this topology, for routing multi terminal cut-nets, their decomposition is first performed to two terminal cut-nets and routing is subsequently performed on two point tracks.

4.2 Generic, direct, two point, multi point interconnect topology (topology 2)

The second topology under consideration is also generic in nature and uses direct connections. However, contrary to topology 1, this topology uses a mixture of two point and multi point tracks where two terminal cut-nets of partitioned design are routed using two point tracks while multi terminal cut-nets are routed using multi point tracks. This topology is off-the-shelf in nature as both two point and multi point tracks are pre-fabricated and they are equally distributed. An example of this topology is shown in Fig. 8b.

4.3 Partially customized, direct, two point, multi point interconnect topology (topology 3)

Similar to topology 2, this topology also uses a combination of two point and multi point tracks for routing of two and multi terminal cut-net signals. However, the number of two point and multi point tracks are not pre-fabricated in this case and they can be customized for a group of applications. Normally, if we know in advance the type or the domain of the designs that we want to prototype using multi-FPGA board, then the customization of two and multi point tracks as per the generic requirement of applications under consideration can result in better frequency results. In this work, for the benchmark suite under consideration, we leverage from the work presented in [26] to determine the best combination of two and multi point tracks for each FPGA board. We call this topology as partially customized as the number of two and multi point tracks are customized for an FPGA board that can prototype a group or domain of applications and the customization is not performed for individual applications. This partial customization may increase the overall set-up time of FPGA-based prototyping but it can result in significant frequency improvement as compared to completely off-the-shelf, generic topologies 1 and 2 (For further details refer Sect. 5).

4.4 Full-custom, direct, two point, multi point interconnect topology (topology 4)

Compared to first three topologies, this topology is fully customized in nature where the best combination of two and multi point tracks is individually determined for each application under consideration. The process of defining a custom board for each application can be explained with the help of example shown in Fig. 9. An example of partitioned benchmark having three partitions with information on two terminal and multi terminal cut-nets is shown in Fig. 9a. It can be seen from Fig. 9a that there are 307 multi terminal cut-nets in the design and total number of cut-nets in the design including multi terminal nets are 1967 which makes percentage of multi terminal nets in the design 15.6%. For custom multi point FPGA board, first multi point tracks are defined using Eq. 4. For example, if total number of available FPGA I/Os are 480, then multi terminal FPGA board after definition of multi point tracks is shown in Fig. 9b. Once multi point tracks for FPGA board are defined, cut-nets and available FPGA I/Os for each partition are updated. For example after multi track assignment of Fig. 9b, remaining FPGA I/Os are 405 and remaining cut-nets for partition 1, 2 and 3 are 781, 1548 and 991 respectively. Once number of two terminal cut-nets for each part are calculated, number of two point tracks between different FPGAs in a custom FPGA board are defined using Eq. 5. In order to determine the number of tracks between different FPGAs, first the part with most number of cut-nets is chosen. In current example, since partition 2 had most number of cut-nets, the number of tracks between partition 2 and 3 are defined as $879 * 405/1548 = 230$ (see Fig. 10a) which further make number of tracks between FPGA 1 and 2 equal to 175. Finally, number of cut-nets between partition 1 and 3 are same, however, FPGA with smaller available I/Os dictates number of tracks between FPGA 1 and 3 which is equal to 175. Final multi, two point custom FPGA board for the example design shown in Fig. 9a is given in Fig. 10b. Here, a simple example of three FPGA board is discussed. However, the mechanism is generic in nature and Eqs. 4 and 5 can be used to define custom boards for any number of FPGAs.

$$Tracks(multipoint) = \frac{multi\ cut\ nets * (FPGA\ I/Os)}{all\ cut\ nets} \tag{4}$$

$$Tracks(x, y) = \frac{cut\ nets(x, y) * (Available\ I/Os\ on\ X)}{cut\ nets\ on\ Part\ X} \tag{5}$$

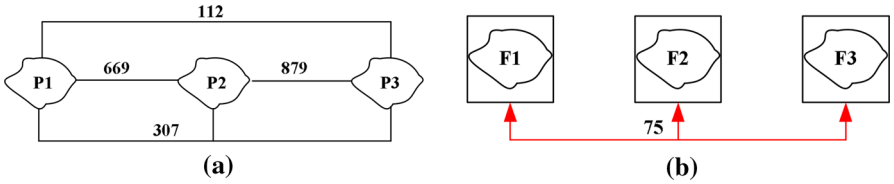

Fig. 9 **a** Partitioned design with two terminal and multi terminal cut-nets; **b** Multi point track assignment

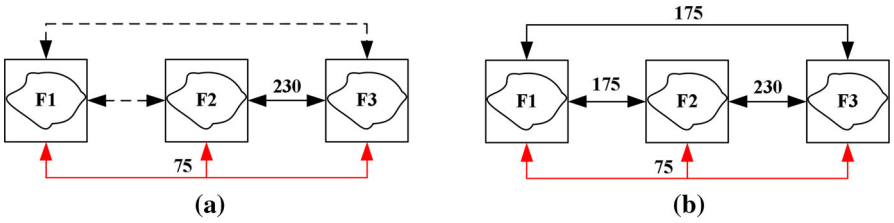

Fig. 10 **a** Multi point and two point track assignment; **b** completed multi, two point tracks assignment for multi-FPGA custom board (Topology 4)

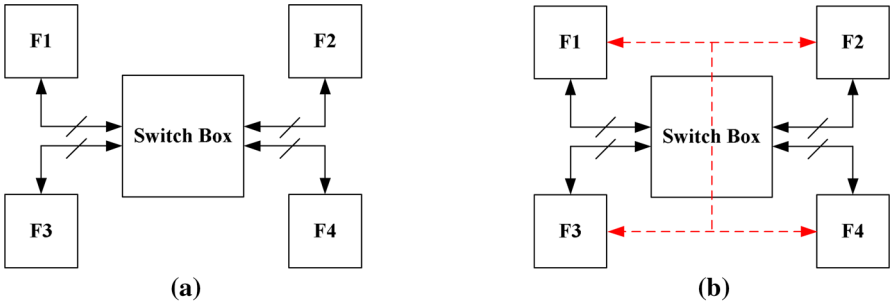

Fig. 11 **a** FPGA board with switch based connection only (Topology 5); **b** FPGA board with a mixture of switch based and multi point connection (Topology 6)

4.5 Generic, switch-based only interconnect topology (topology 5)

Instead of using direct two- or multi point tracks for interconnect between different FPGAs of the board, this topology uses a configurable switch for the interconnect. The switch used for the interconnect is a partial cross bar and it uses disjoint [30] connection pattern. This topology is also generic in nature as the switch box used for FPGA interconnect is pre-fabricated. However, this switch is configurable in nature and the connections can be programmed as per the routing requirement of different applications that are being prototyped on the board. The aim behind using this topology is to give the FPGA interconnect some flexibility as higher flexibility is expected to produce good multiplexing ratio results. Figure 11a shows an abstract level view of topology 5. It can be seen from this figure that all the connections between different FPGAs on board are controlled through a configurable switch box. Internal architectural details of the switch box are also shown in Fig. 12. It can be seen from this figure that the switch box uses disjoint connection pattern and the connections between different I/Os of the FPGAs can be configured by changing the SRAM values of the multiplexers shown in Fig. 12.

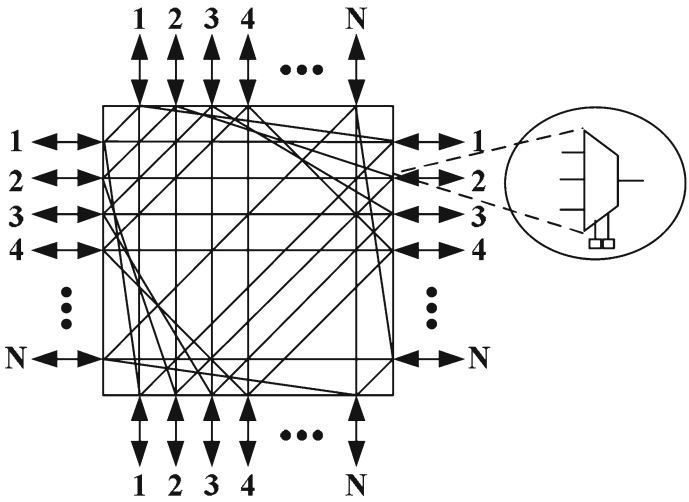

Fig. 12 Internal structure of reconfigurable switch box

Table 2 Qualitative comparison of six interconnect topologies

Interconnect Topology	Flexibility	Interconnect nature	Unit Price	Setup time
Topology 1	High	Generic	Low	Low
Topology 2	High	Generic	Low	Low
Topology 3	Moderate	Partial custom	Moderate	Moderate
Topology 4	Low	Full custom	High	Large
Topology 5	High	Generic	Low	Low
Topology 6	Moderate	Partial custom	Moderate	Moderate

4.6 Generic, switch-based, multi point interconnect topology (topology 6)

Just like topology 1, topology 5 routes all the cut-nets in point-point manner through its configurable switch. For large FPGA boards, this kind of interconnect topology might give good results for designs containing only two terminal cut-nets because of the increased switch box flexibility. But practical applications contain a large number of multi terminal cut-nets and their routing using topology 5 requires their decomposition into two terminal cut-nets. This scenario will eventually lead to a rise in number of intermediate hops; hence resulting in a poor execution speed at the end. In order to cope with this issue, we propose topology 6 where two terminal nets are routed through switch box whereas multi terminal cut-nets are routed through direct multi point tracks. Figure 11b shows the abstract level architectural description of topology 6. This topology is also generic in nature where switch box connections are prefabricated and they can be configured as per routing requirement of target design. This topology is slightly less flexible as certain percentage of its routing resources are dedicated to direct multi point tracks which may lead to higher multiplexing ratio. However, for the same reason, the number of hops are reduced which may eventually result in better execution speed as compared to topology 5.

Table 3 Benchmark details

S. no	Benchmark name	No. of ALMs	No. of Registers
1	AES	52,450	38,230
2	CPU2×2×1	53,032	40,191
3	CPU2×2×2	59,489	45,812
4	CPU2×2×3	69,611	50,579
5	CPU2×2×4	78,257	55,764
6	CPU2×2×5	206,527	162,355
7	CPU2×2×6	213,234	167,553
8	CPU2×2×7	223,900	172,753
9	CPU2×2×8	713,852	583,463
10	CPU4×4×2	735,324	584,324

In this section, we have given detailed overview of six interconnect topologies under consideration. A qualitative analysis of different interconnect topologies is also presented in Table 2 [35]. It can be seen from this table that all the topologies have their plus and negative points. In the next section, we present the frequency and routing time results of these topologies for different FPGA boards. Based on the characteristics of topologies given in Table 2 and the results obtained in next section, we will present a comprehensive analysis and give the best tradeoff in terms of topology selection for a particular FPGA board.

5 Results and analysis

In this section, we present detailed experimental results that are obtained through the flow described in Sect. 3. For experimentation, we use six interconnect topologies and frequency results for each topology are obtained using four different FPGA boards where the number of FPGAs on board varies from three to six. To explore different interconnect topologies and FPGA boards, we use ten large benchmarks. Architecture of these benchmarks is already explained in Sect. 2 and their generation process is given in Sect. 3. An overview of the generated benchmarks is given in Table 3. These are multi-clustered MPSoCs where the number of clusters varies from 4 to 16 (product of first two digits in the benchmark name column of Table 3) and number of processors in each cluster vary from 1 to 8 (the third digit in the benchmark name column of Table 3). These benchmarks are generated through DSX tool described in Sect. 2. Results presented in this section are mainly divided into two parts: first we present the mux ratio, frequency comparison results for six interconnect topologies and then we present routing time analysis of each interconnect topology. Finally, a frequency–time tradeoff analysis is also presented at the end of this section.

5.1 MUX ratio and frequency comparison results

As discussed earlier, for experimentation, we use four different FPGA boards. For each board, benchmarks under consideration are synthesized, partitioned, routed using flow described in Sect. 3 and optimized mux ratio is determined at the end of the flow. Normalized mux ratio comparison results of six interconnect topologies using three FPGA board are shown in Fig. 13. Before we continue with the analysis of the results, it is important to mention here that for topology 1, 2, and 5, the number of two point and multi point tracks are distributed in a

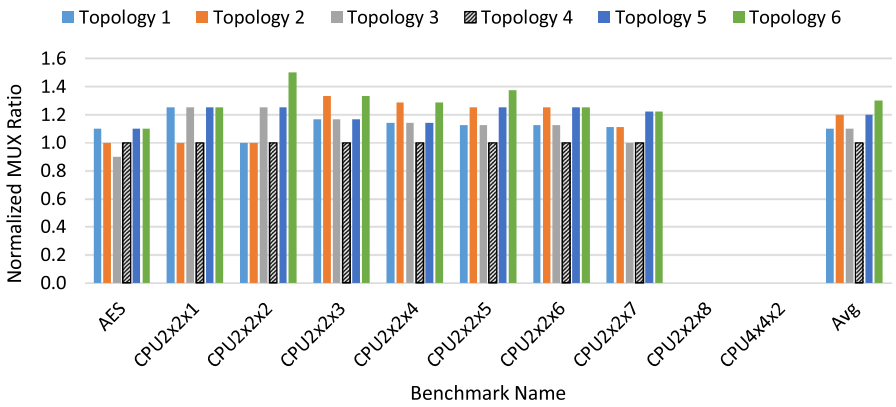

Fig. 13 Normalized mux ratio comparison of six interconnect topologies for three FPGA boards

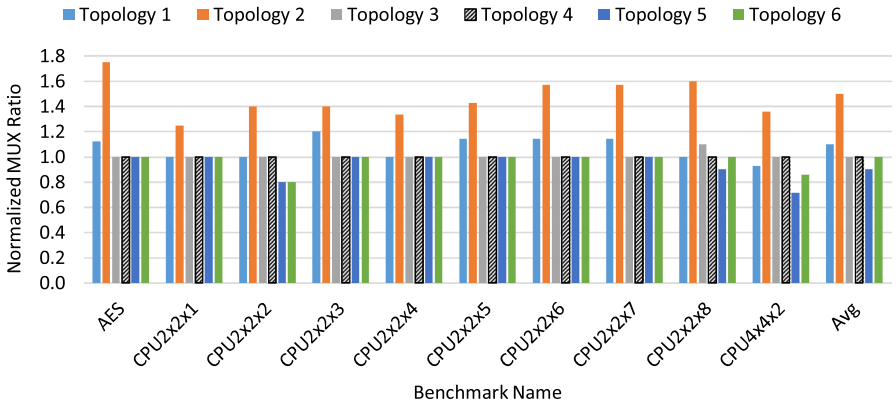

Fig. 14 Normalized mux ratio comparison of six interconnect topologies for four FPGA boards

balanced way for each FPGA board. For topology 3 and 6 we use a combination of multi point and two point tracks that gives best frequency results [26] for the set of ten benchmarks under consideration. For fully customized topology 4, the mechanism to determine the number of multi point and two point tracks for each application has been explained in detail in Sect. 4. In Fig. 13, used benchmarks names are given along the X-axis while corresponding normalized mux ratio results are shown along the Y-axis. It can be seen from this figure that the two largest benchmarks give no results as they are too large to fit in the logic capacity of three Stratix 5 target FPGAs. Furthermore, it can also be noted from this figure that we use topology 4 as our reference topology as it is the only fully customized topology among the six topologies and it also gives, on average, the best mux ratio results for three FPGA board. It can be seen from Fig. 13 that topology 4 gives either equal or better mux ratio results for all the benchmarks and on average, it gives 9, 17, 9, 17, and 23% better mux ratio results as compared to topologies 1, 2, 3, 5, and 6 respectively. Contrary to the expectation, topology 5 gives poor mux ratio results as compared to topology 4. This is because of the fact that when the number of FPGAs on board are small, the flexibility of the switch box is also limited and it results in higher multiplexing ratio results.

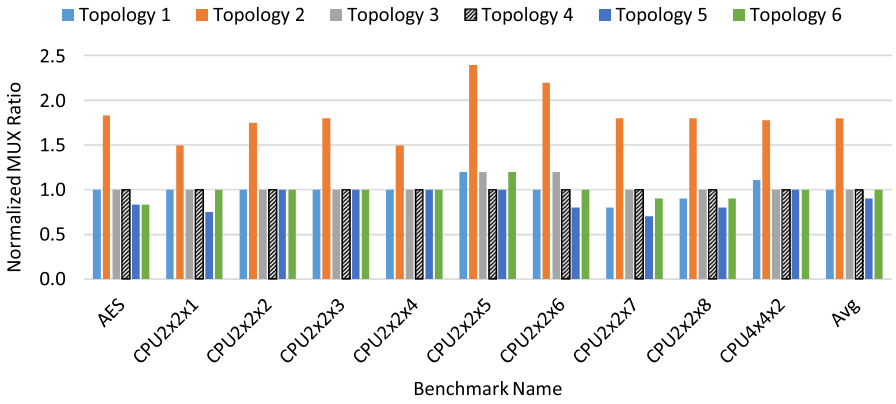

Fig. 15 Normalized mux ratio comparison of six interconnect topologies for five FPGA boards

Normalized mux ratio comparison results for four FPGA board are shown in Fig. 14. It can be seen from this figure that for four FPGA board, we are able to route even the two largest benchmarks of the suite. For four FPGA board, topology 5 gives best normalized mux ratio results and compared to topology 4, on average, it gives 9% smaller mux ratio. This is because of the fact that when the number of FPGAs on board increase, the flexibility of switch box also increases; hence resulting in overall improved mux ratio results for topology 5. It is interesting to note here that mux ratio results of topology 5 are better than topology 6 although topology 6 also uses switch box. The reason is that some percentage of total tracks in topology 6 are reserved for multi point connections that reduces the number of tracks available for switch box; hence resulting in smaller flexibility and higher mux ratio as compared to topology 5. It can also be noted from Fig. 14 that topology 2 produces the worst mux ratio results and compared to topology 4, on average, it requires 50% more mux ratio. This is because of the fact that in topology 2, number of two and multi point tracks are distributed in an equivalent manner and routing requirement of the set of applications under consideration is neither explored (as in topology 3, 6) nor customized (as in topology 4). So, in our case, multi terminal cut-nets are much more as compared to multi point tracks and they dictate the overall multiplexing ratio. This issue was not as vivid in case of three FPGA boards because the number of partitions were small. However, the trend for topology 2 observed in four FPGA board continues to larger FPGA boards as well and mux ratio gap becomes even larger for five and six FPGA boards.

Normalized mux ratio results for five FPGA board are shown in Fig. 15. It can be seen from this figure that for five out of ten benchmarks under consideration, topology 5 gives better results as compared to topology 4 and for remaining 5 benchmarks, it gives same mux ratio results as topology 4. Trend observed for topology 2 in four FPGA board continues for five FPGA board as well. The mux ratio results for topology 2 become even worse and topology 2, on average, requires 80% higher mux ratio as compared to topology 4. As far as topology 1, 3 and 6 are concerned, they give same average mux ratio results as topology 4. Mux ratio results for six FPGA board are shown in Fig. 16. Trends observed for 3, 4, 5 FPGA board uphold here as well. Topology 5 produces the best and topology 2 gives the worst results and the gap between topology 2 and 4 increases to 100%.

Although results presented in Figs. 13, 14, 15 and 16 give some insight into the characteristics of six topologies, they do not give the complete picture about the efficiency of each topology. For this purpose, we present next the system frequency results of each topology for

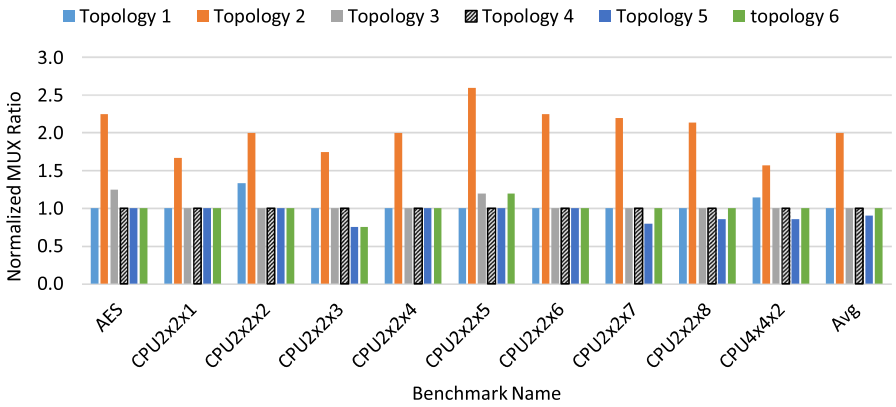

Fig. 16 Normalized mux ratio comparison of six interconnect topologies for six FPGA boards

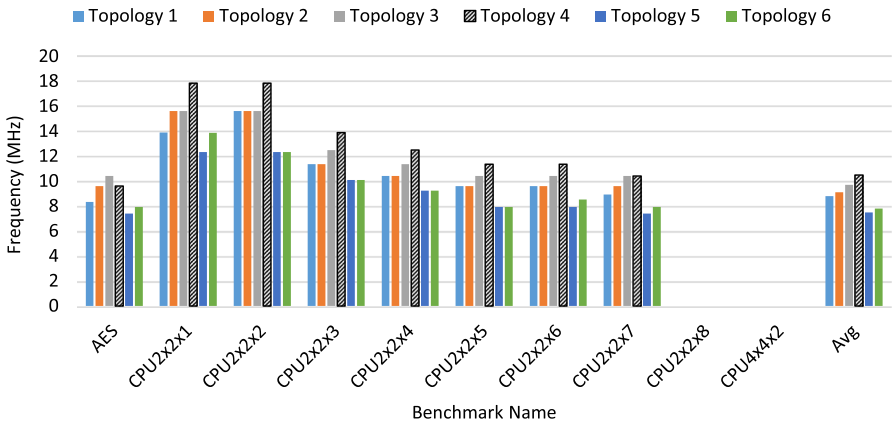

Fig. 17 Frequency comparison results of six interconnect topologies for three FPGA boards

four different FPGA boards where system frequency for individual benchmarks is calculated using Eq. 3. Frequency comparison results for three FPGA board are shown in Fig. 17. Just like mux ratio results, we choose topology 4 as our reference topology. The if_freq value for topology 4 is same as for first three topologies. However, better average mux ratio (see Fig. 13) and reduced number of hops lead to an average frequency gain of 19.3, 15.3, and 8.2% for topology 4 as compared to topology 1, 2 and 3 respectively. For topologies 5 and 6 the if_freq value is reduced to 111 MHz because of presence of switch box. This fact combined with poor mux ratio and increased hop value results in average frequency gain of 40 and 35.2% for topology 4 as compared to topology 5 and 6 respectively.

Frequency comparison results for four FPGA board are shown in Fig. 18. As shown in Fig. 14, topology 2 gives, on average, the worst mux ratio results and as a result this topology gives, on average, the worst sys_freq average frequency results (see Fig. 18) for four FPGA board. Furthermore, it should also be noted that although topology 5 gives the best mux ratio results for four FPGA board, its sys_freq results are as poor as topology 2. This is because of the reduced if_freq of topology 5 along with increased number of hops. It can also be noted from Fig. 18 that topology 3 gives almost same average frequency result as topology 4 and, on average, it is only 3% slower as compared to topology 4. As far as topology 6 is concerned,

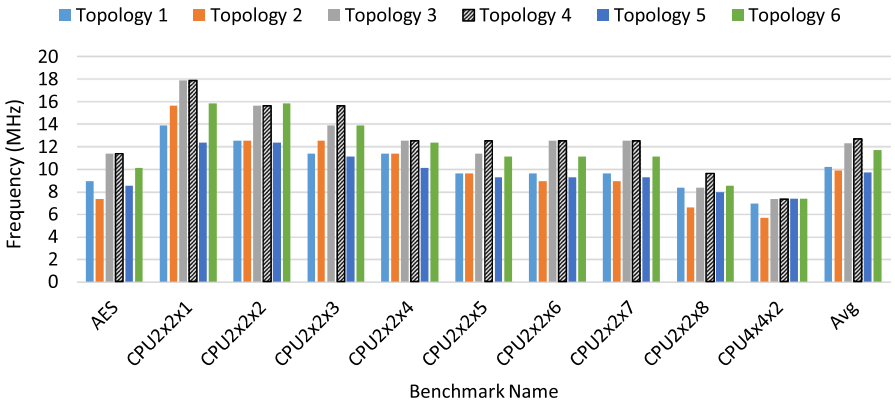

Fig. 18 Frequency comparison results of six interconnect topologies for four FPGA boards

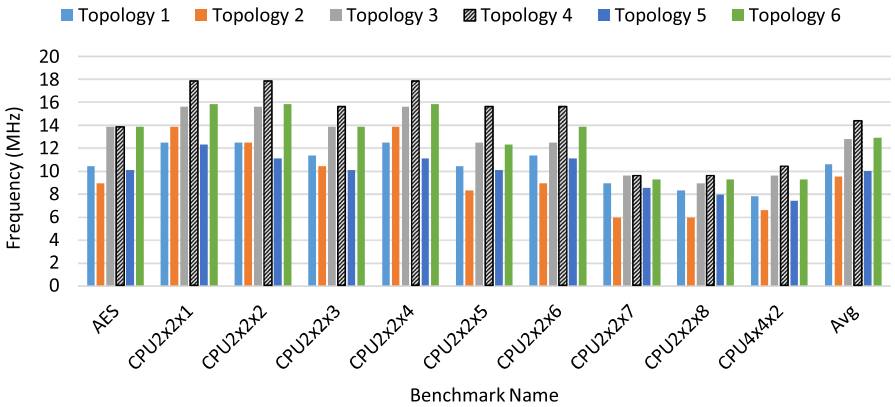

Fig. 19 Frequency comparison results of six interconnect topologies for five FPGA boards

despite smaller if_freq value, its improved mux ratio results reduce the frequency gap from 35.2% in three FPGA board to 10% in case of four FPGA board.

Frequency comparison results for five and six FPGA boards are shown in Figs. 19 and 20 respectively. It can be seen from these figures that trends observed in four FPGA board continue for five and six FPGA boards as well. For five FPGA board, topology 2 gives, on average, the worst results and topology 5 despite its better mux ratio results (see Fig. 15) gives second worst frequency results. The average gap between topology 4, 3 is at 11.1% whereas between topology 4, 6, it is at 10.4%. Frequency comparison results for six FPGA board (see Fig. 20) show similar trend where the gap between topology 4, 3 is increased to 15.1% and the gap between topology 4 and 6 remains at 10.4%.

From the results presented in Figs. 13, 14, 15, 16, 17, 18, 19, and 20, it can be concluded that mux ratio alone is not a correct indicator of the efficiency of an interconnect topology and in some cases it can be even misleading. So for complete performance picture, it is always better to have the system frequency comparison results for the topologies under consideration. Moreover, it can be seen from these figures that fully customized topology 4 gives the best frequency results for all the boards under consideration. It can also be concluded that topologies 3, 6 whose two point and multi point track combinations are

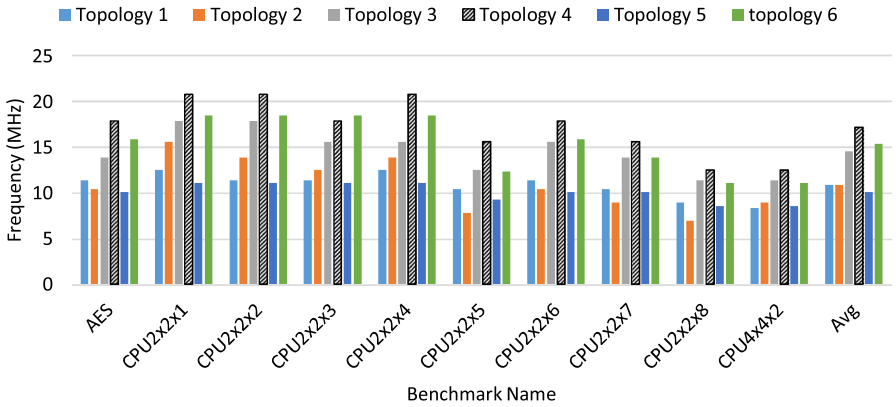

Fig. 20 Frequency comparison results of six interconnect topologies for six FPGA boards

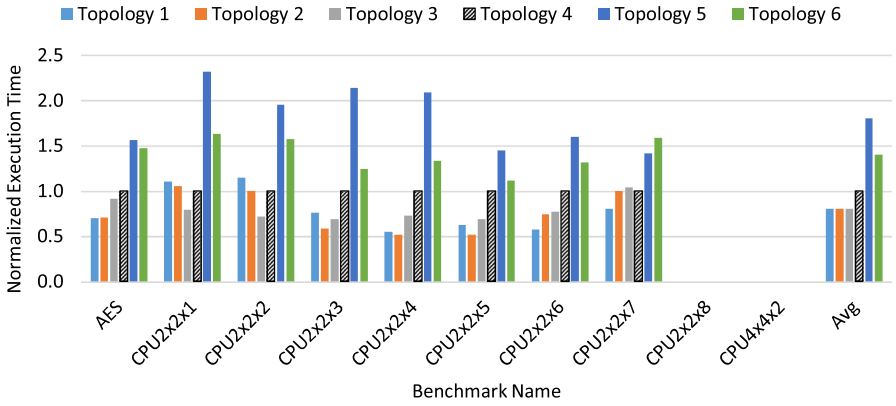

Fig. 21 Routing time comparison results of six interconnect topologies for three FPGA boards

determined through exploration mechanism of [26] are not far behind topology 4 and despite being generic in nature they give results comparable to topology 4. This is because of the fact that these two topologies leverage from the exploration results of [26]. It can also be noted from these figures that topology 3 gives better results for smaller boards like 3 and 4 FPGA board whereas topology 6 gives better results for bigger boards (i.e. 5 and 6 FPGA board) Lastly, it can be seen from these figures that generic topologies 1, 2, 5 give the worst frequency results for all FPGA boards under consideration.

5.2 Routing time comparison results

Apart from frequency comparison, we also perform routing time comparison between six interconnect topologies. By the routing time, we mean the time taken to perform inter-FPGA routing step as the rest of the steps do not make any difference for different interconnect topologies. Normalized routing time comparison results for three FPGA board are shown in Fig. 21. These results are obtained by running the inter-FPGA routing tool on a 64 bit linux server with 16 cores each running at 3 GHz and having 64 GB RAM. It can be seen from this figure that for almost all benchmarks, topology 2 gives the best routing time results

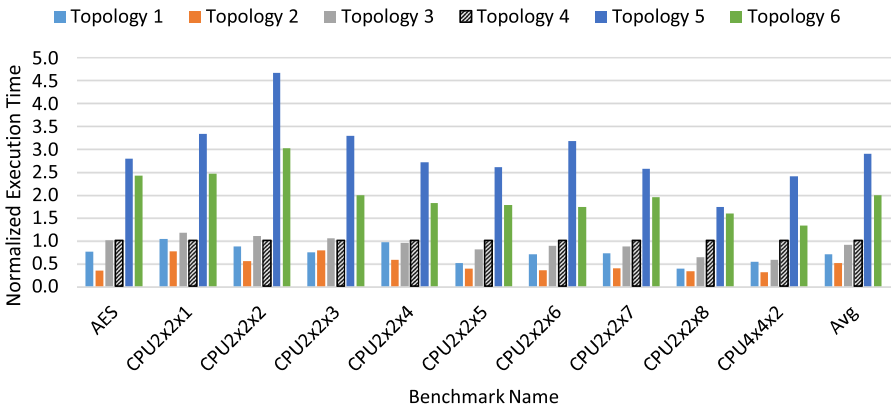

Fig. 22 Routing time comparison results of six interconnect topologies for four FPGA boards

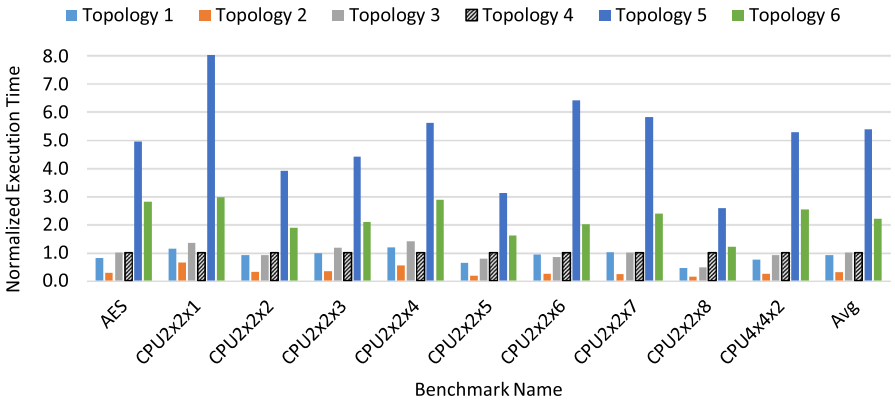

Fig. 23 Routing time comparison results of six interconnect topologies for five FPGA boards

while topology 5 gives the worst routing time results. Since topology 2 is based on generic two point and multi point direct connections only, its hyper graph is simple and Pathfinder quickly finds the conflict free solution. Furthermore, its mux ratio is normally much higher as compared to other topologies; hence it requires fewer routing rounds eventually resulting in smaller routing times. As far as topology 5 is concerned, it is purely based on switch box. The reconfigurability in switch makes the hyper graph large and complex and in turn Pathfinder requires more time to find a conflict free solution. The complexities of the graphs of topologies 1, 3, 4, and 6 are in between these two extremes so they give routing time results in between topology 1 and 5. A comparison between the reference topology (i.e. topology 4) and other topologies under consideration shows that for a three FPGA board, on average, topology 1, 2, 3 require 30, 30, 20% less time while topology 5, 6 require 80, 40% more time respectively. Similar trend continues for larger FPGA boards as well and the routing time gap between the best and worst topologies increases even further. For example, for four FPGA board (see Fig. 22), compared to reference topology, topology 2 requires 50% less routing time while topology 5 requires 2.9x more time respectively. For five and six FPGA boards, this gap is even higher. For five FPGA boards, the gap between topology 4 and 2, 5

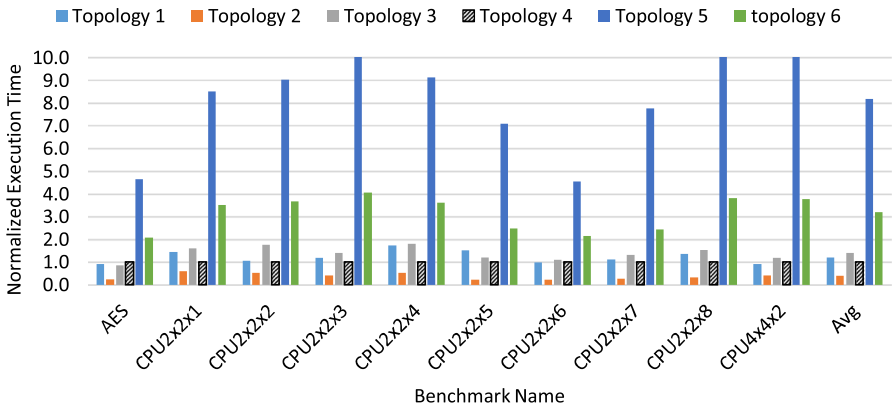

Fig. 24 Routing time comparison results of six interconnect topologies for six FPGA boards

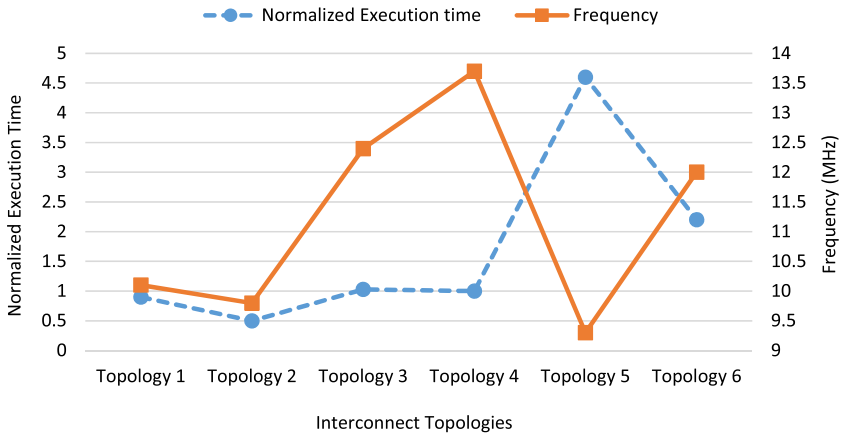

Fig. 25 Average frequency and routing time comparison between different interconnect topologies

is 70%, 5.4 times (see Fig. 23), and for six FPGA board, it is 70% and 8.2× respectively (see Fig. 24).

A global summary of the average frequency and routing time comparison between four interconnect topologies is presented in Fig. 25. It can be seen from this figure that for four FPGA boards, topology 2 gives the best routing time results while topology 4 gives the best frequency results. Further analysis of these topologies shows that, compared to topology 2, topologies 1, 3, 4, 5 and 6 require 1.8×, 2×, 2×, 9.2× and 4.4× more average time. Frequency comparison of topology 4 with topology 1, 2, 3, 5, and 6 shows that these topologies give, on average, 26.2, 28.5, 9.5, 32.1 and 12.4% smaller frequencies respectively. frequency–time analysis of different interconnect topologies shows that topology 4 is the best from performance perspective with good routing time results. However, the issue with this topology is that it is fully customized with no flexibility and opting for this topology can result in large setup time delays and high price (see Table 2). So, topology 4 is preferred when the performance of the design is principle constraint and price, setup time are not an issue. Now, if we look at topology 3 and 6, both of them have moderate setup time and both are partially customized. Further among these two topologies, if we look at the frequency results (see

Figs. 17, 18, 19, 20), topology 3 produces better average frequency results as compared to topology 6 for smaller FPGA boards (i.e. three FPGA board, four FPGA board). However, for five and six FPGA boards, topology 6 gives better frequency results. So, it can be concluded that for smaller FPGA boards, topology 3 should be preferred and for larger FPGA boards topology 6 can be used as both offer moderate price and setup time. However, topology 4 is the best choice for the designs having high performance as their principle constraint.

6 Conclusion

Multi-FPGA based pre-silicon verification of digital systems has gained tremendous popularity over the past few years. In this work, a comprehensive exploration and comparison of six inter-FPGA interconnect topologies for multi-FPGA systems is presented. This exploration focuses on the evaluation of performance and routing time metrics of different interconnect topologies under consideration. For exploration, ten large benchmarks are generated, partitioned and routed on four different FPGA boards. For interconnect topology exploration, a generic inter-FPGA routing tool is designed. Results obtained through experimentation reveal that fully customized topology (i.e. topology 4) gives best average frequency results for all the FPGA boards under consideration. Frequency comparison between different topologies reveals that as compared to topology 4, topology 1, 2, 3, 5 and 6 give, on average, 26.2, 28.5, 9.5, 32.1, and 12.4% smaller frequencies respectively. Routing time comparison between different topologies is also performed. Experimental results reveal that generic interconnect topology using a balanced, predetermined, hybrid combination of two and multi point tracks (i.e. topology 2) gives best routing time results. Its comparison with topologies 1, 3, 4, 5 and 6 reveals that they require 1.8×, 2×, 2×, 9.2×, and 4.4× more average time respectively. Finally, frequency–time tradeoff analysis along with board flexibility, setup time and unit cost shows that partially customized topology (i.e. topology 3) gives the best performance results for smaller FPGA boards and topology 6 gives the best results for larger FPGA boards. Both these topologies offer comparable performance along with better flexibility and smaller setup time as compared to topology 4.

Acknowledgements This research work is done through the financial support of Systematic and Bpifrance. Authors would also like to thank Zied Marrakchi and Hayder Mrabet from Flexras Technologies for their suggestions and guidance to enhance the quality of this work.

References

1. Santarini M (2005) Asic prototyping: Make versus buy, EDN, vol 11
2. Sigenics: Custom asic calculator, http://www.sigenics.com/page/custom-asic-cost-calculator (2017)
3. AMD, http://techreport.com/news/13721/chip-problem-limits-supply-of-quad-core-opterons (2007)
4. Pentium, https://en.wikipedia.org/wiki/pentium_fdiv_bug (1994)
5. Huang C-Y, Yin Y-F, Hsu C-J, Huang TB, Chang T-M (2011) Soc hw/sw verification and validation. In: 16th Asia and South Pacific design automation conference (ASP-DAC 2011). IEEE, pp 297–300
6. Cadence, https://www.cadence.com/content/cadence-www/global/en_us/home/tools/system-design-and-verification/simulation-and-testbench-verification/incisive-enterprise-simulator.html (2017)
7. Graphics M (2017) https://www.mentor.com/products/fv/modelsim/
8. Vcs: A functional veriifcation solution by synopsys, http://www.synopsys.com/Tools/Verification/FunctionalVerification/Pages/VCS.aspx (2017)
9. Series CP (2017) http://www.cadence.com/products/sd/palladium_xp_series/pages/default.aspx
10. Zebu-server asic emulator by synopsys, http://www.synopsys.com/tools/verification/hardware-verification/emulation/Pages/default.aspx (2017)

11. Veloce MG (2017) https://www.mentor.com/products/fv/emulation-systems/
12. Kuon I, Rose J (2010) Quantifying and exploring the gap between FPGAs and ASICs, vol 1, Springer, Ed. Springer US
13. Babb J, Tessier R, Dahl M, Hanono S, Hoki D, Agarwal A (1997) Logic emulation with virtual wires. IEEE Trans Comput-Aided Des Integr Circuits Syst 16(6):609–626
14. Krupnova H (2004) Mapping multi-million gate SoCs on FPGAs: industrial methodology and experience. In: Design, automation and test in europe conference and exhibition, 2004. proceedings, vol 2. pp 1236–1241
15. Asaad S, Bellofatto R, Brezzo B, Haymes C, Kapur M, Parker B, Roewer T, Saha P, Takken T, Tierno J (2012) A cycle-accurate, cycle-reproducible multi-FPGA system for accelerating multi-core processor simulation. In: Proceedings of the ACM/SIGDA international symposium on field programmable gate arrays, ser. FPGA '12. ACM, New York, pp 153–162. [Online]. https://doi.org/10.1145/2145694.2145720
16. Yang S (1991) Logic synthesis and optimization benchmarks user guide, version 3.0
17. Farooq U, Parvez H, Mehrez H, Marrakchi Z (2012) A new heterogeneous tree-based application specific FPGA and its comparison with mesh-based application specific FPGA. Microprocess Microsyst 36(8):588–605. [Online]. https://doi.org/10.1016/j.micpro.2012.06.012
18. Stroobandt D, Verplaetse P, Van Campenhout J (2000) Generating synthetic benchmark circuits for evaluating cad tools. IEEE Trans Comput-Aided Des Integr Circuits Syst 19(9):1011–1022
19. Pouillon N, Greiner A (2010) Soc lib project. [Online]. https://www.asim.lip6.fr/trac/dsx/
20. Panades IM, Greiner A, Sheibanyrad A (2006) A low cost network-on-chip with guaranteed service well suited to the gals approach. In: Nano-networks and workshops, 2006. NanoNet '06. 1st international conference on, 1–5
21. Certify, http://www.synopsys.com/Prototyping/FPGABasedPrototyping/Pages/Certify.aspx (2017)
22. Inagi M, Takashima Y, Nakamura Y (2009) Globally optimal time-multiplexing in inter-FPGA connections for accelerating multi-FPGA systems. In: Field programmable logic and applications, 2009. FPL 2009. International conference on, pp 212–217
23. Turki M, Marrakchi Z, Mehrez H, Abid M (2013) 9th International Symposium, ARC 2013, Los Angeles, CA, USA, March 25–27, 2013. Springer, ch. Iterative routing algorithm of inter-FPGA signals for multi-FPGA prototyping platform, 201–217
24. DiniGroup, http://www.dinigroup.com/ (2017)
25. Tang Q, Mehrez H, Tuna M (Oct 2013) Routing algorithm for multi-FPGA based systems using multi-point physical tracks. In: Rapid system prototyping (RSP), 2013 international symposium on, pp 2–8
26. Farooq U, Chotin-Avot R, Azeem M, Ravoson M, Turki M, Mehrez H (2016) Inter-FPGA routing environment for performance exploration of multi-FPGA systems. In: Proceedings of the 27th international symposium on rapid system prototyping, ser. RSP '16. New York: ACM, pp 107–113
27. Kim C, Shin H (1996) A performance-driven logic emulation system: FPGA network design and performance-driven partitioning. IEEE Trans Comput Aided Des Integr Circuits Syst 15(5):560–568
28. Hauck S, Borriello G, Ebeling C (1994) Mesh routing topologies for multi-FPGA systems. In: Proceedings 1994 IEEE international conference on computer design: VLSI in computers and processors, 170–177
29. Khalid MAS, Rose J (2000) A novel and efficient routing architecture for multi-FPGA systems. IEEE Trans Very Large Scale Integr VLSI Syst 8(1):30–39
30. Kuon I, Tessier R, Rose J (2008) Fpga architecture: survey and challenges. Found Trends Electron Des Autom 2:135–253
31. McMurchie L, Ebeling C (1995) Pathfinder: a negotiation-based performance-driven router for FPGAs. In: ACM international symposium on field-programmable gate arrays. ACM Press, New York, 111–117
32. Dijkstra EW (1959) A note on two problems in connexion with graphs. Numer Math 1(1):269–271
33. Xilinx, http://www.xilinx.com (2017)
34. Altera, http://www.altera.com (2017)
35. Amos D, Lesea A, Richter R (2011) FPGA-based prototyping methodology manual: best practices in design-for-prototyping, Synopsys, Ed. Synopsys