Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2010, Article ID 390545, 15 pages
doi:10.1155/2010/390545

Research Article

Flexible Interconnection Network for Dynamically and Partially Reconfigurable Architectures

Ludovic Devaux, Sana Ben Sassi, Sebastien Pillement, Daniel Chillet, and Didier Demigny

IRISA, Université de Rennes, 6 rue de Kerampont, BP 80518, 22302 Lannion, France

Correspondence should be addressed to Ludovic Devaux, ludovic.devaux@irisa.fr

Received 1 July 2009; Accepted 30 November 2009

Academic Editor: Elías Todorovich

The dynamic and partial reconfiguration of FPGAs enables the dynamic placement in reconfigurable zones of the tasks that describe an application. However, the dynamic management of the tasks impacts the communications since tasks are not present in the FPGA during all computation time. So, the task manager should ensure the allocation of each new task and their interconnection which is performed by a flexible interconnection network. In this article, various communication architectures, in particular interconnection networks, are studied. Each architecture is evaluated with respect to its suitability for the paradigm of the dynamic and partial reconfiguration in FPGA implementations. This study leads us to propose the DRAFT network that supports the communication constraints into the context of dynamic reconfiguration. We also present DRAGOON, the automatic generator of networks, which allows to implement and to simulate the DRAFT topology. Finally, DRAFT and the two most popular Networks-on-Chip are implemented in several configurations using DRAGOON, and compared considering real implementation results.

1. Introduction

Steady technological evolutions, increasingly complex applications, can be supported by reconfigurable architectures. This is particularly true into the framework of complex signal processing applications. However, when the number of tasks constituting an application exceeds the available hardware resources, designers have two options: increasing the number of resources (which increase the complexity of the systems) or implementing only the tasks that should be executed at a given time. In the latter case, tasks are swapped at the end of their execution by freeing logical resources for the others. However, this concept is relevant only if a new task can be implemented instead of a former one without disrupting the execution of other tasks. This concept of hardware preemption of the tasks is called *Dynamic and Partial Reconfiguration* (DPR).

The objective of the FOSFOR research project (Flexible Operating System FOr Reconfigurable devices) [1] is to specify and to implement an *Operating System* (OS) that provides an abstraction of technology for future applications. For this purpose, the FOSFOR OS operates the DPR in order to support complex applications that cannot be statically implemented due to physical restrictions. Several services of the OS are implemented in hardware whereas others are computed in software. Physical implementation of some services allows the OS to efficiently manage hardware and software tasks. In this direction, the FOSFOR project focuses on the hardware implementation of the main services: the task placer and scheduler, the memory manager, and the communication service.

In this work, the communication service is investigated. The objective is to define and to implement in an FPGA a generic interconnection architecture which should support the diversity of applications and the dynamic management of the tasks. For this purpose, the architecture takes into account the constraints imposed by the DPR. Through the physical interconnection architecture and its control, the communication service provides a flexible way for transferring data between every *Communicating Element* (CE) in an FPGA. Since an application task can be implemented in hardware or processed in software by a hardware processor,

CEs are defined as the hardware elements which exchange data. So, a CE can be the hardware implementation of a task (static or dynamic), a shared element (memory, input/output), or a hardware processor running software tasks.

The paper is organized as follows. Assumptions induced by the DPR that should be supported by an interconnection architecture are presented in Section 2. Then, current interconnection architectures are detailed and reviewed in Section 3. The interconnection network called *Dynamic Reconfiguration Adapted Fat-Tree* (DRAFT), which is specifically designed to support the DPR requirements in FPGAs, is presented in Section 4. *Dynamically Reconfigurable Architectures compliant Generator and simulatOr Of Network* (DRAGOON), the automatic generator of networks supporting the DRAFT topology is introduced in Section 5. Then, the comparison between DRAFT and the two more popular *Networks-on-Chip* (NoC) topologies is presented with respect to implementation costs and network performances in Section 6. Finally, an implementation of the DRAFT network into the framework of an application from the FOSFOR project is detailed in Section 7.

2. Assumptions on Considered Interconnection Architectures

Keeping in view of a large range of applications, an interconnection architecture should support several constraints into the framework of an implementation in FPGAs. Current applications are very complex and their task graphs exhibit a large degree of parallelism. Thus, from the interconnection point of view, the architecture must provide the possibility to realize several communications in parallel. Furthermore, dynamic placement and scheduling of CEs in an FPGA require a high level of flexibility, since the placement and the scheduling of the CEs are dynamic. So, neither the location of CEs nor the data traffic (uniform, all to one, etc.) can be predicted at compile time. These requirements of flexibility should be considered by the network topology, the routing protocol, and the available network performances (bandwidth and latency). An application is typically split into dynamic and static tasks, and there are no reason for every task to be implemented in homogeneous hardware CEs. This point differs from approaches like [2] where CEs are supposed to be homogeneous so that the interconnection topology can be dynamically reconfigured into an FPGA in order to fit the application. Heterogeneous hardware CEs are considered in this work. Hence, a single interconnection topology compliant with this heterogeneity is preferred to dynamically reconfigurable topologies considering their placement constraints as the placement of the CEs themselves.

Considering current FPGAs, Altera devices provide abundant programmable resources but do not support the partial reconfiguration of everyone of them [3, 4]. Atmel series AT40K5AL to AT40K40AL are compliant with the DPR of their resources but do not provide more than 50 K gates which are few when considering complex applications

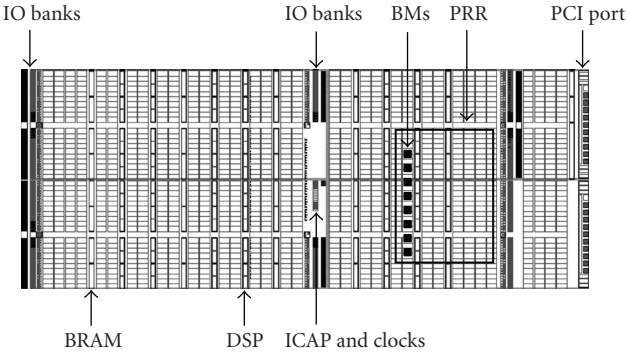

Figure 1: View of a Xilinx Virtex5 5VSX50T FPGA captured from the PlanAhead Xilinx software [7].

[5]. Hence, Xilinx FPGAs (especially the Virtex 2Pro, Virtex 4, Virtex 5, and Virtex 6) are the only ones that both provide sufficient programmable resources to implement complex applications and support the DPR paradigm [6]. Xilinx architectures offer a column-based distribution of their resources, as shown Figure 1. In a Virtex family (except in Virtex 2Pro), the center column is very specific because it contains not only Input/Output (IO) banks, but also clock managers and DPR ports (ICAP). So, despite their column-based structure, FPGAs are very heterogeneous. Consequently, the interconnection should support the heterogeneity of the FPGAs.

Xilinx DPR takes place in specific reconfigurable regions called *Partially Reconfigurable Regions* (PRRs), which hence constitute the dynamic parts of a system. Dynamic CEs are allocated in these regions. Static CEs (static hardware tasks, memories, processors, etc.) are implemented all around the PRRs and constitute the static part of the system. Communications between dynamic and static regions are performed through interfaces called Bus Macros (BMs) in Xilinx architectures [8]. Since PRRs and BMs are defined statically at compile time, every CE is connected to the interconnection architecture through static interfaces (despite the dynamic locations of the CEs). So, the interconnection architecture can be static if it provides sufficient flexibility to support the DPR paradigm.

3. Interconnection Architectures: State of the Art

In this section, several interconnection architectures, suitable for supporting the paradigm of the DPR, are analyzed and evaluated. The main constraints are the parallelism of the communications, the flexibility, and the compliance with the connection of heterogeneous CEs in heterogeneous FPGAs.

3.1. Bus-Based Architectures. Buses are interconnection architectures which are simple to implement and control, while requiring few hardware resources.

3.1.1. Main-Bus-Based Architectures. One of the first bus-based approaches designed for DPR was Core Unifier [9].

Core Unifier is a tool that allows connecting on the fly dynamic CEs on a bus. For this, the tool adds 3-states buffers to the bus on which dynamic CEs are connected. However, at compile time, Core Unifier needs the knowledge of when and where the CEs should be allocated. So, this approach is not compliant with systems in which scheduling and placement are dynamic.

The Bus-Macro-based architecture [10] and Recobus [11] are the most recent bus-based dynamic interconnection architectures. The communication interfaces of the Recobus approach require less logical resources than the Bus-Macro-based architecture. However, both approaches are based over a horizontal bus connecting vertically placed CEs. CEs are implemented using all the logical resources of a column. So, those approaches are only compliant with the 1D placement (CEs are allocated using several columns of resources of the FPGA) of the dynamic CEs. It is the major limitation of these architectures because it leads to a waste of hardware resources, depending on the size of the CEs. Furthermore, current FPGAs (except the Xilinx Virtex 2Pro series) are compliant with the 2D placement (rectangular region based) of CEs. So, none of these approaches seems suitable to implement a system using DPR in latest FPGAs.

Another architecture is the HoneyComb [12], which is a static interconnection architecture connecting static CEs but differing from other bus-based structures. This network, made of regular hexagonal cells linked together through bidirectional buses, allows data routing. HoneyComb offers flexibility at the data path level while using few hardware resources. However, the regular structure is not compliant with current FPGAs nor with dynamic and heterogeneous CEs. So, Honeycomb brings more flexibility than other bus-based architectures but it does not support the DPR paradigm due to its regular structure.

3.1.2. Limitations of Buses. Buses exhibit several drawbacks when considering an application using the DPR. A major constraint for the bus-based interconnection architecture is the necessity to support several data transfers in parallel. One bus supports only a single transfer at a time. There are two solutions for this problem. One is to use several communication lines (multiple buses [13]); the other is to split a bus into a set of independent segments interconnected through several bridges (GALS approach [14]). However, either solution implies an increase of the required hardware resources: the connection of two buses generates a need of buffers (synchronous or asynchronous FIFOs). In order to estimate the size of these buffers, the knowledge of the data traffic is required to avoid bottleneck risks. In a dynamic system where the locations of CEs and the data traffic are unknown at compile time, buffers should be designed to support the worst cases of communication. For instance, the RMBoC approach [15] uses multiple and segmented buses for DPR compliance. However, despite the estimation of buffer sizes, this approach requires a lot of hardware resources to connect the CEs.

To date, the bus-based architectures have not been implemented efficiently to answer the DPR paradigm. Poor flexibility and scalability also with placement limited to 1D are the main drawbacks of bus-based approaches which motivated many researchers to consider the *Network-on-Chip* (NoC) paradigm [16].

3.2. Network-on-Chip Based Architectures. A NoC can be seen as a set of routers, links and network interfaces. The communication topologies and routing protocols (including data flow control, scheduling policies, etc.) are key domains of research [17]. Nowadays, there are a lot of NoCs which can be classified in a few basic families, depending on their topology. Hence, we shall concentrate on the meshes, the application-dependent topologies, the rings and the trees (Figure 2).

3.2.1. Mesh Topology. A mesh topology offers a simple connection scheme (Figure 2(a)), based on a matrix of routers interconnecting regularly implemented CEs. Many meshes were implemented such as HERMES [18]. If the number of connected CEs is N and if D is the radix of the mesh, then the number of routers needed to build a square mesh is

$$R_{\text{mesh}} = \left\lceil \sqrt{N} \right\rceil^2 = D^2. \tag{1}$$

Whereas the number of needed communication links is

$$L_{\text{mesh}} = N + 2\left(D^2 - D\right). \tag{2}$$

The connection of heterogeneous CEs to a regular matrix based network may be problematic. A solution consists in considering that tasks are implemented using one or several homogeneous CEs [19] which are interconnected regularly by the mesh. This solution is used, for example, by the DyNoC approach [20]. Unfortunately, logical resources could be wasted when a small task is executed by a large CE. When a large task is implemented using several CEs, some logical resources are also wasted depending on the granularity of the CEs (Figure 3).

When a task is implemented using several CEs, then CEs can communicate in two ways. The first one is to consider that the communications between the CEs are performed through the network. However, this solution constrains the design of the tasks because each internal communication should be designed to be performed by the network. So, with this solution, the complexity of the tasks increases and the number of required resources too. Thus, this solution is impractical for designers. Another solution is to consider that a task is implemented with dedicated links between the CEs. However, the compliance between the task and the technology becomes difficult due to the limited (and reduced) number of available communication links.

A truly regular mesh is based on the assumption that the FPGA is intrinsically homogeneous. So, every CE presents the same hardware properties for task implementation. Considering the heterogeneous structure of available FPGAs, it seems quite difficult to implement a truly regular mesh. Finally, another problem of the mesh structure lies in the connection of shared elements like memories or IOs. Indeed,

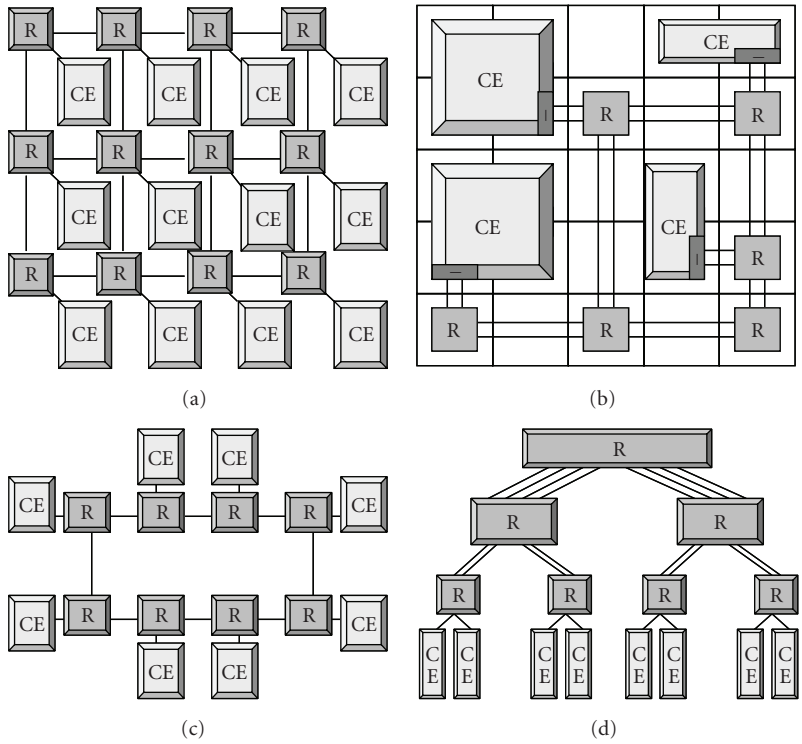

Figure 2: Network topologies: (a) mesh, (b) application-dependent topology, (c) ring, and (d) tree. Routers are denoted with "R" and communicating elements with "CE".

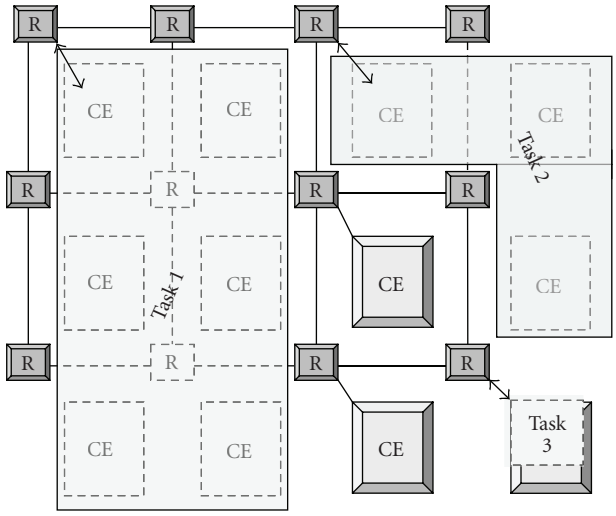

Figure 3: Implementation of three different tasks (executed by one or several CEs) using a mesh network.

the communication requirements between tasks (hardware or software) and shared elements induce the creation of hotspots, which could increase the likelihood of livelock and deadlock.

3.2.2. Application-Dependent Topology. CoNoChi (Figure 2(b)) shows an example of network which topology differs depending on the application [21]. In this approach, the FPGA is divided into regular dynamically reconfigurable regions, with each being defined dynamically as a router, a set of interconnection links, or a CE.

Despite many advantages, the concept of dynamic topology is impractical since PRRs are statically defined in current FPGAs. Hence, their use for link implementation implies wasting a lot of logical resources. Moreover, considering the resulting topology of CoNoChi which differs with the application, the design time can be important to obtain optimal network performances. Indeed, the routing protocol should take into account the network topology which is application-dependent. Keeping in mind these drawbacks, static and generic topologies are preferred to dynamic ones in order to limit the design times and to increase the predictable nature of the network performances.

3.2.3. Ring Topology. Rings (Figure 2(c)) are presently used by industrials like IBM with the Cell Broadband Engine [22]. This architecture can be very efficient. However, in some applications, it offers lower performances than a mesh [23]. Indeed, the available bandwidth depends on the characteristics of the routers and on the number of interconnected CEs. Another drawback of the rings lies in the dependence between latencies and CE locations. So, in the context of the DPR, rings should be interesting if bandwidth did not decreased when the number of connected CEs increases, and furthermore if the latency was not dependent with the placement of the CEs.

3.2.4. Fat-Tree Topology. Trees, and more precisely fat-trees (Figure 2(d)), are indirect interconnection networks. Some routers are only used for data transfers and do not connect directly the CEs. A fat-tree is based on complete binary tree. Every CE is connected to a router located at the base-level of the tree. Each hierarchical level of the fat-tree is linked to upper and lower levels through bidirectional links [23]. The main characteristic of a fat-tree is the aggregative bandwidth (offered by all the links located at the base of the tree) which remains constant between each hierarchical level all the way to the root. A fat-tree offers many advantages compared with other topologies, two of which are a large bandwidth and a low latency [24]. A fat-tree can also simulate every other topology at the cost of the appropriate control [25].

Since every CE is connected to a base-level router of the fat-tree, they are not distributed into the network structure. This point is important because CEs can be heterogeneous from the resource consumption point of view. Indeed, the resource heterogeneity does not impact the transfer times like in a mesh. Furthermore, the structure of the tree does not need to be implemented regularly. So, a fat-tree is compliant with current FPGAs.

Thanks to its constant bandwidth between every hierarchical level, the fat-tree avoids deadlock risks which exist in other topologies without an appropriate control [26].

However, a fat-tree needs many logical resources for routing purpose when the number of connected CEs increases significantly. If the number of connected CEs is N and the number of communication ports for each router is k, then the number of routers in a fat-tree [26] is

$$R_{\text{fat-tree}} = \frac{2N}{k}\left(\log_{k/2} N\right). \tag{3}$$

In this formula, assuming that the fat-tree is complete in terms of connected CEs, N is expressed by $N = 2^x$ where x is an integer and $x \geq 1$. When N does not match the previous formula, designers should build the network considering the admissible value of N just higher in order to keep the complete tree-based structure of the network. The number of connection links needed by the fat-tree topology [26] is

$$L_{\text{fat-tree}} = N\left(\log_{k/2} N\right). \tag{4}$$

To limit the resource consumption, the XGFT [27] allows the connection of several CEs to only one communication port of a router. Indeed, the fat-tree connects several sets of CEs, which are interconnected by a bus. This approach is very interesting in the context of a static application because this topology optimizes the number of connected CEs in comparison with used resources for routing purpose. However, the XGFT is not optimal into the framework of the DPR. Indeed, the sets of CEs have the same drawbacks as bus-based architectures (control time, one communication at a time without multiple buses, etc.).

3.3. Summary of the Interconnection Architectures. The compliance between presented interconnections and the constraints of the DPR paradigm are summarized in Table 1.

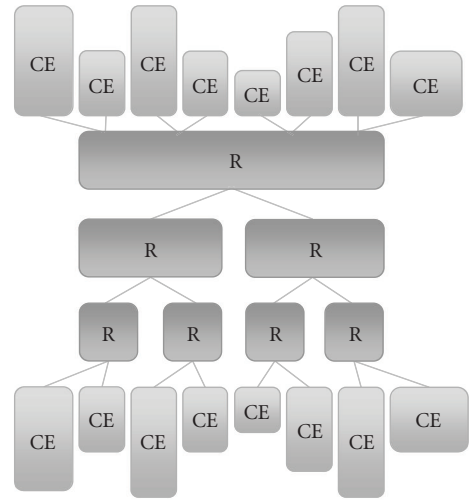

Figure 4: The DRAFT topology. Communicating elements "CEs" are connected to the root and base-levels of a fat-tree network.

From Table 1, a fat-tree is best adapted to the DPR paradigm and applicative requirements. However, its resource consumption remains the main drawback for an implementation into an FPGA. Thereby, this study of current interconnections leads to the *Dynamic Reconfiguration Adapted Fat-Tree* (DRAFT) network.

4. Flexible Interconnection: DRAFT

From the comparison of current NoCs, the fat-tree appears to be the most suitable interconnection architecture to support the DPR paradigm and applicative requirements. DRAFT is a fat-tree-based network whose main characteristic lies in the reduction of needed resources for routing purpose. Like a fat-tree, DRAFT interconnects several CEs, which could be implementations of hardware tasks (static or dynamic), processors running software tasks, and shared elements like shared memories.

4.1. DRAFT Topology. The concept proposed in DRAFT is to directly connect half of the CEs to the root-level routers of a fat-tree (Figure 4). This concept reduces a lot the number of hardware resources used for routing purpose compared with the number of connected CEs. Indeed, for the same number of connected CEs, the number of routers is divided by two when compared with the fat-tree topology.

Concerning network performances, the distances between the root and base-level CEs are constant whatever their locations. So, the minimal latencies of these communications are constant. However, for the effectiveness of this topology, it is necessary that the CEs connected to the root communicate only with the base-level connected ones. This assumption avoids the creation of hotspots in the root-level router. Communications between the CEs connected to the root would require additional hierarchical levels (leading to the fat-tree topology) in order not to increase the load on the root-level router. However, the base-level connected CEs can

Table 1: Compliance between current interconnection structures and DPR constraints.

DPR requirements	Buses	Mesh	CoNoChi	Ring	Fat-tree
Dynamic scheduling	NO	OK	OK	OK	OK
2D placement	NO	OK	OK	OK	OK
Heterogeneous CEs	OK	NO	OK	OK	OK
Heterogeneous FPGAs	OK	NO	NO	OK	OK
Resource consumption	OK	OK	NO	OK	NO
Routing flexibility	NO	OK	—	NO	OK
Communication parallelism	NO	OK	OK	OK	OK
Bandwidth	—	NO	—	NO	OK
Latencies	NO	NO	—	NO	OK

freely communicate with every other CE. This assumption is very important and while it is observed, there is no limitation concerning the nature of the CEs. A designer is free to connect its shared elements to the root-level of DRAFT, but also some hardware tasks (static or dynamic) or even processors. So, at the cost of this assumption, DRAFT is completely flexible.

4.2. Hardware Requirements. The hardware resources required by DRAFT should be considered first. If the number of connected CEs is N and the number of communication ports for each router is k, then the number of routers in DRAFT is

$$R_{\text{DRAFT}} = \frac{N}{k}\left(\left(\log_{k/2} N\right) - 1\right). \quad (5)$$

In this formula, assuming that DRAFT is complete in terms of connected CEs, N should be in the form of $N = 2^x$ where x is an integer and $x \geq 2$. If the number of CEs does not match previous formula at design time, then designers should build the network considering the just higher admissible value of N. Similarly, the number of connection links needed by DRAFT is

$$L_{\text{DRAFT}} = \frac{N}{2}\left(\log_{k/2} N\right). \quad (6)$$

From this last formula, DRAFT uses two times less connection links than a fat-tree. This is an advantage for the implementation in current FPGAs.

There are several ways to see a fat-tree and so is DRAFT. Thereby, the two fat-trees presented in Figures 4 and 5 have the same properties regarding the CEs. Indeed, a router in Figure 4 can be broken up into a set of several unitary routers called fat-node (Figure 5). This permits to build the network by using a single router type, which is generically defined, and makes the automatic generation of DRAFT easier. However, the latter structure is not fully compliant with the connexion of the CEs to the root-level, since there is only one admissible data path between base- and root-level CEs. The router-based structure (Figure 4) is more flexible due to multiple data paths. If a communication link is already used, data can be transferred through another one to the same destination. This traffic adaptive approach is not possible in the fat-node-based structure for the communications between CEs from the root and base-levels. However, in both structures, a traffic adaptive routing can be implemented for the communications between two base-level connected CEs. So, the fat-node-based structure is less flexible than the router-based one but more generic. Furthermore, it allows to demonstrate the viability of DRAFT even if it constrains the data paths. In a first time, the focus is over the fat-node-based structure due to the generic routers. In future works, the structure will be switched to the router-based one in order to support applications requiring multiple data paths between elements from the base and root-levels.

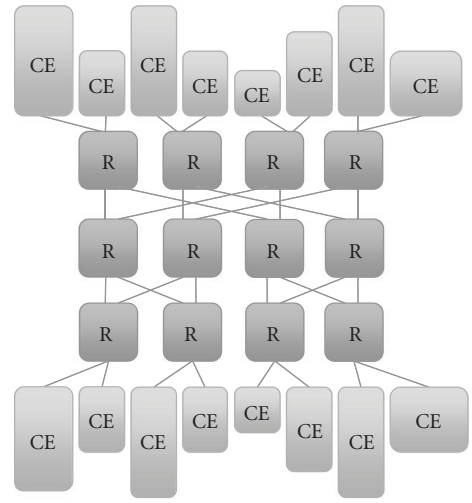

Figure 5: Fat-node view of DRAFT architecture. A fat-node is a set of unitary routers.

4.3. Principles of Connection of Various CEs. Since many applications require data transfers between static and dynamic CEs, they should be connected to DRAFT. For this purpose, designers can connect the static CEs directly to the routers, or with a bus-based sub-network (multiple buses e.g.,) connected to a single router, like for the XGFT network. In the latter structure, while CEs are static, the sub-network can be optimally designed for their communication needs. Then, the sub-network and its connected CEs (statically implemented tasks, processors, shared elements, etc.) are viewed by DRAFT as a single CE. On the other hand,

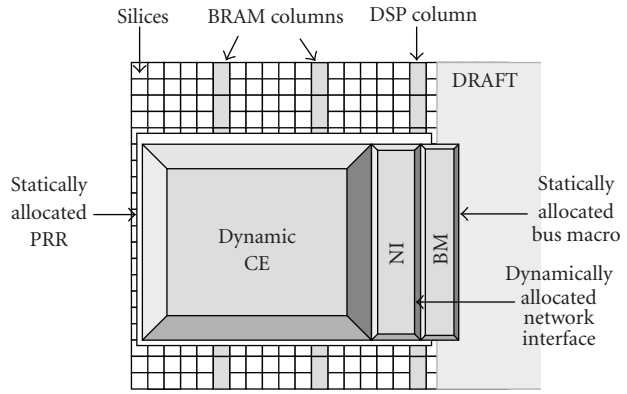

Figure 6: PRR receiving a dynamic CE connected to DRAFT through the dynamic NI and the static BMs.

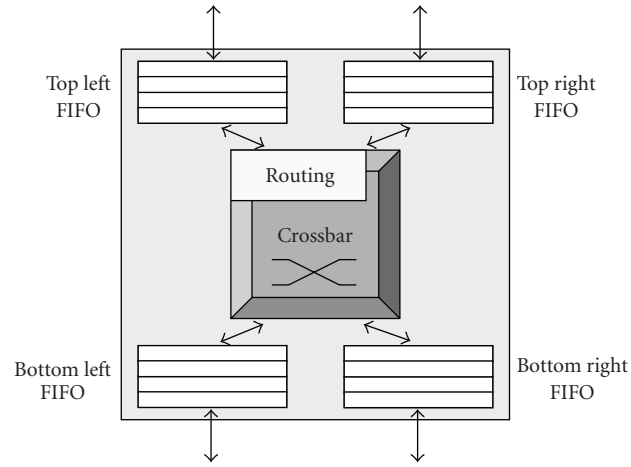

Figure 7: View of the DRAFT unitary router architecture.

the static CEs which do not exchange data with dynamic ones should be interconnected through a separate network optimally designed for this purpose. Similarly, the shared elements which do not communicate with dynamic CEs but only with static ones should be connected to a sub-network rather than to DRAFT. These principles are advised to reduce the size of DRAFT and so its resource consumption. Doing so, DRAFT can be seen as an independent core of network or even as an *Intellectual Property* (IP) block connecting each part of the application, while providing the flexibility required by the DPR paradigm.

For applications using the DPR, DRAFT does not directly connect the tasks through their network interfaces, but through the Bus Macros (BMs). So, BMs are the interfaces between DRAFT and the dynamic CEs (including their network interfaces (NIs)) as presented in Figure 6.

This concept of dynamically reconfigurable NIs is important because they can be designed optimally for their corresponding CEs. This allows to reduce the hardware cost of the NIs when a CE does not have the same interface as the others. So during the dynamic reconfiguration of a given PRR, DRAFT interface remains the same even if the newly allocated CE presents a specific interface. So, this concept makes DRAFT more generic and more flexible considering the location of the CEs.

4.4. Hardware Characteristics of the Routers. The router architecture (Figure 7) is based over four bidirectional communication ports, each including an asynchronous FIFO. FIFO sizes are defined by the designer depending on the flit width (data are fractioned into several small sets of bits called flits), and on the number of flits to store, according to the data flow and the livelock management protocol (credit based, priority, round robin, etc.). A crossbar, controlled by a routing manager which protocol is based on a Turn-Back algorithm [28], performs the routing of data.

The next destination of a message is computed by each router receiving it. The decision is made using several masks over the message source and destination addresses included into the message header. Each router is identified by an internal address which indicates its hierarchical level and its location into this level. Similarly, every CE connected to DRAFT is identified by an internal address which is used to specify the source and the destination of a message. This addressing of the CEs and routers is presented in Figure 8.

Thanks to these addresses, each router uses a hierarchical level dependent mask to determine if a flit should be routed toward an upper hierarchical level in order to reach a different part of the tree. So, each data is routed toward the high until it is able to go down to the desired half (or subpart) of the tree. This lowering routing is directly applied to the CEs connected to the root-level of DRAFT. Destination addresses are sufficient to determine toward which part of the tree a data should be routed. This algorithm, presented in Algorithm 1, provides a minimal distance to the data transfers and the guaranty that there is no deadlock risk [27]. In this algorithm, the source and destination addresses of a data are called, respectively, *CEsrc* and *CEdest*. The *mask* is directly calculated from the *Y* address of the router corresponding to its hierarchical level. As an example, for router X:0010 Y:0001, the corresponding *mask* is 0011. The *Mshift* parameter is the *mask* previously calculated shifted right of one bit set to 1. Thus, in this example, *Mshift* is 1001. Similarly, the *RXshift* is calculated from the X address of the router shifted left of one bit, that is, 0100 in the example.

In order to keep static the DRAFT architecture into the framework of the DPR, addressing of the routers and CEs is generic. However, since many CEs are dynamic, it constrains the designer to make sure that the task placer/scheduler keeps up-to-date a routing table. This table is essential for the network interfaces of the CEs to make the correspondence between the internal addresses and the physical elements (implemented task, memory, etc.).

4.5. Implementation of DRAFT in a Xilinx FPGA. DRAFT placement is important in the conception of a system using the DPR, because it impacts the use of the reconfigurable resources as well as the network performances. Thus, the concept presented in Figure 9 is to implement DRAFT as a central column into the FPGA. This concept is particularly

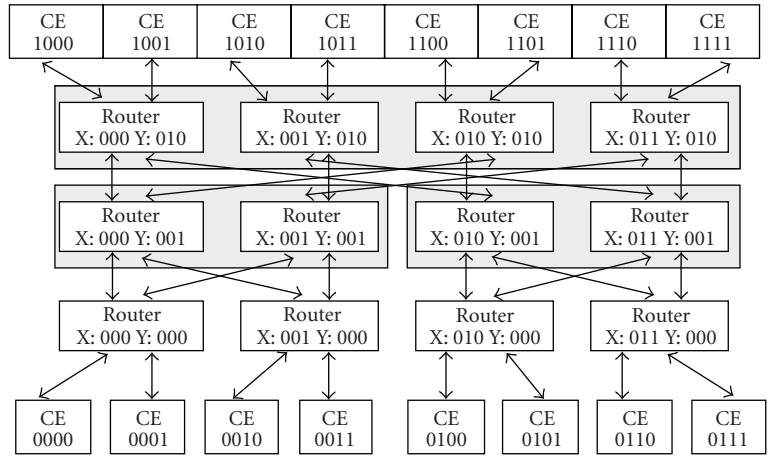

Figure 8: Generic addressing of the CEs and routers depending on their hierarchical levels and their locations into these levels.

```
IF("data from port West or South")
// data arrived from a lower level
        IF((RXshift AND mask) XOR (CEdest AND mask) = 0)
        // is the aimed subpart of DRAFT reachable from this router ?
                "Yes: routing toward lower level"
        ELSE
                "No: routing toward upper level depending on CEsrc"
        END IF
ELSE
// data arrived from an upper level
        IF((RXshift AND Mshift) XOR (CEdest AND Mshift) = 0)
        // which subpart of DRAFT is the data destination ?
                "routing toward the right side"
        ELSE
                "routing toward the left side"
        END IF
END IF
```

Algorithm 1: Routing algorithm implemented into DRAFT routers.

adapted to current technologies supporting the DPR: Xilinx Virtex 4, Virtex 5, and Virtex 6 FPGAs. CEs are implemented into both halves of the FPGA with the static elements of the application (processor, etc.). Since DRAFT is not distributed into the FPGA, the designer is not constrained by the network for the definition of the CEs in terms of sizes and locations. This is an advantage for the implementation of heterogeneous dynamic CEs. Thus, the implementation of DRAFT is fully compliant with current technology and the DPR requirements.

In present technology, PRRs and BMs are defined statically, but there is no physical obstacle to make them dynamic. The limitation is only due to the design tools which do not support the dynamic definition of the partially reconfigurable regions. Consequently, if the design software allows the definition of dynamically locatable PRRs, then every base- (or roots) level router should be reachable from both halves of the FPGA. Doing so, the dynamic relocation of a PRR from one half of the FPGA to the other will be supported.

In Xilinx FPGAs, shared IOs should be located into the central IO bank column. Similarly, it is recommended to locate the shared memories into the BRAM columns the nearest of the central column. Thereby, the communications between the CEs and the shared elements encounter a minimal latency.

5. Presentation of DRAGOON

In this section, the design software called *Dynamically Reconfigurable Architecture compliant Generator and simulatOr Of Network* (DRAGOON) is presented. DRAGOON is a conception environment specifically designed to generate and to simulate the DRAFT topology. It is inspired from ATLAS which was developed to support Hermes NoCs [18]. DRAGOON is also compliant with the fat-tree topology. The conception flow provided by DRAGOON is illustrated Figure 10, in which every step corresponds to a tool.

The NoC generation tool produces the VHDL description of DRAFT and test benches written on SystemC,

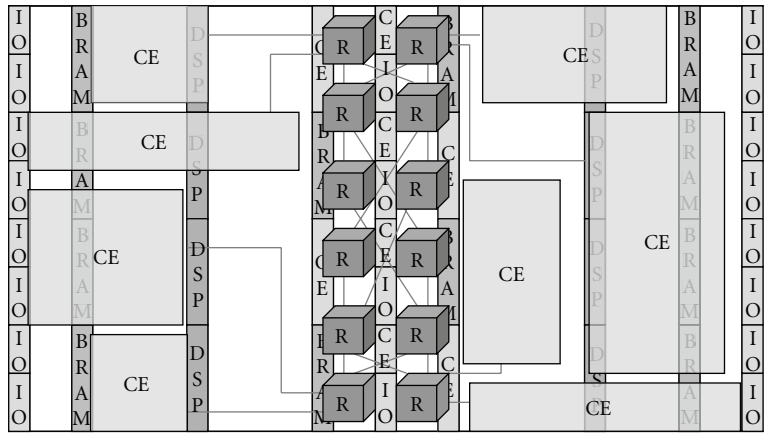

Figure 9: Implementation of DRAFT as a central column interconnecting CEs which are located into both halves of the FPGA.

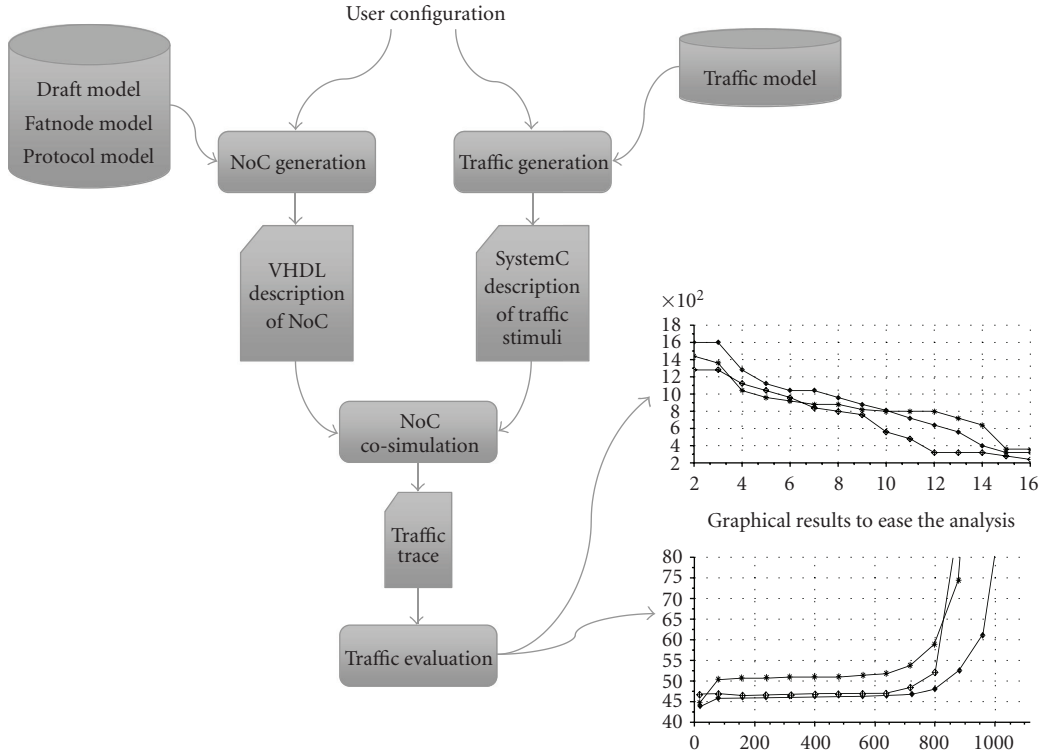

Figure 10: Conception flow of DRAGOON.

according to the configuration chosen by the user. This latter is able to choose the network dimension in terms of connected CEs, and also the flit width or the buffer depth. The flit width parameter determines the length of the data (fractioned into unitary flits) exchanged through the network. Flit width can be set to 16, 32, or 64 bits. The buffer depth indicates how many flits can be stored into one of the four FIFOs of a router. So, a router can store 4, 8, 16, or 32 flits into each of its 4 ports. The number of virtual channels can also be chosen by the user. Using virtual channels, a router is able to support several communications in parallel at the cost of hardware resources. Concerning the data traffic, the type of flow control (credit based or handshake) and the scheduling policy (round robin or priority) are parameterizable. So, the NoC generation tool allows generating DRAFT networks adapted to the requirements of many applications.

The traffic generation tool produces different data traffics (uniform, normal, pareto on/off). Each traffic simulates an application supported by DRAFT. A uniform traffic simulates applications using a constant data flow like video processing. A normal traffic is provided by applications using data dependency like pattern recognition or target tracking. Finally, the pareto on/off traffic simulates the communications between a task and a shared memory where data is transmitted using a burst mode (period of

uninterrupted data transmission followed by a period of silence). Furthermore, the traffic generation tool allows the designer to simulate several configurations of the connected CEs. Indeed, the frequencies of the CEs, the targets of data (random or specific), the number of packets to send, and the number of flits in a packet are parameterizable. Designer can also specify the transmission rate of each CE. Using all these parameters, the traffic generator builds input files containing data to be transmitted through the network.

The simulation tool invokes an external VHDL simulator (ModelSim). This simulator was chosen because it supports mixed VHDL and SystemC. Thus, the simulation tool uses the description of the NoC and the generated traffic. This traffic is injected into DRAFT during the simulation phase, which is concluded when the output files are generated.

The evaluation tool provides the interpretation of the results thanks to the previously generated output files. Results are analyzed and presented through graphics and analysis reports. Network performances like latency and throughput are the main results provided by the evaluation tool.

6. Implementation Results

In this section, DRAFT implementation results are presented. Thanks to the automatic network generator (DRAGOON), DRAFT is compared with the mesh and the fat-tree topologies. This comparison takes into account the use of hardware resources and the network performances. The impacts of the NoC and traffic parameters over hardware and network characteristics are also presented. Hardware resources are obtained thanks to Xilinx ISE 9.2i tool chain [29], and network performances are measured through ModelSim 9.5c [30] and presented thanks to DRAGOON. From this comparison, the viability and the effectiveness of DRAFT are demonstrated.

For a fair comparison of the three different network topologies, some hypothesis must be considered. Every topology is implemented for maximal network performances. So, both DRAFT and fat-tree architectures are based over a complete binary tree whatever the number of connected CEs. Similarly, every implemented mesh presents a square matrix based structure whatever the number of connected CEs. The mesh topology is implemented and simulated using ATLAS while DRAFT and fat-tree topologies are provided by DRAGOON. Since the fat-tree and DRAFT are generated with a fat-node-based structure, the three networks are implemented with the same router architectures and without virtual channels. The routing algorithm is the only component which differs from a topology to another one, so the topologies are compared independently of their router architecture. Every router is clocked at 100 MHz.

6.1. Hardware Resources. DRAFT is defined as a network which supports the DPR requirements and minimizes the hardware resource consumption. In this part, implementation results are investigated, and presented in Figure 11. For this purpose, each router was implemented with a flit width of 32 bits and a buffer depth of 4 flits.

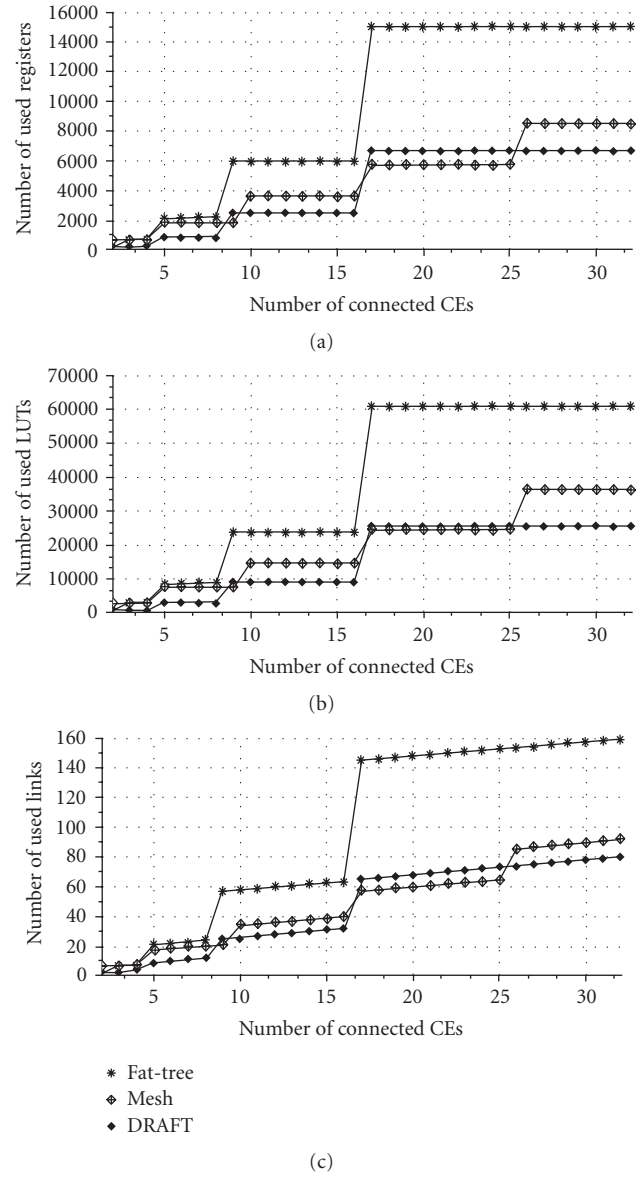

Figure 11: Number of registers (a), LUTs (b), and links (c) used for DRAFT, fat-tree, and mesh implementations in a Xilinx Virtex5 depending on the number of connected CEs.

From these implementation results, as expected, a fat-tree needs more hardware resources and more communication links than every other topology. However, using the root-level connection of the CEs, DRAFT needs less hardware resources and less communication links than a mesh when the number of connected CEs is less than 16. When this number increases, resource consumptions are very close but DRAFT needs less resources when the number of connected CEs becomes close to a power of 2. However, DRAFT outnumbers mesh resources in small ranges like from 17 to 25 connected CEs. This point is due to the assumption that DRAFT is implemented as a complete

binary tree and the mesh as a square matrix. Hence, DRAFT needs 32 routers to connect from 17 to 32 CEs while a mesh needs only 25 routers to connect from 17 to 25 CEs. Then, this latter needs 36 routers to connect from 26 to 32 routers. The number of routing wires increasingly becomes a limiting factor in FPGAs, so the fact that DRAFT requires less links than a mesh is an important matter for the designers. Using the DRAFT network, designers have more communication links available for the implementation of their tasks.

6.2. Network Performances. Usually, latency and throughput results are presented depending on the injection rate. The injection rate corresponds to the percentage of the maximal bandwidth which is used to send data from the CEs point of view. DRAFT and fat-tree topologies have two CEs connected to each base- (or roots) level router. However, in a mesh, only one CE is connected to a router, which directly impacts the injection rate. As an example, a 100% injection rate in a mesh corresponds to a data rate of 3200 Mbit/s per CE whereas it corresponds to 1600 Mbit/s per CE in DRAFT or in a fat-tree. The three topologies are simulated connecting 8 CEs with 32 bits flit width and a buffer depth of 4 flits. The frequency of each CE is 100 MHz, and data are sent with a uniform repartition of their sources and destinations. Results are presented in Figure 12. The number of data to transmit is calculated depending on the injection rates for 1ms of continuous data injection.

For a fair comparison of the three topologies, a comparison of the latencies and throughputs depending on the transfer rates is presented in Figure 13.

From these results, if the NoCs are compared depending on the injection rates calculated from their maximal bandwidth, the mesh topology saturates with an injection rate of 25%. The fat-tree and DRAFT saturate, respectively, around 55% and 60% of injection rates. This point is important because it shows that DRAFT supports a higher injection rate than every other topology. Furthermore, with the comparisons depending on the transfer rates, it appears that DRAFT provides a lower average latency while supporting higher transfer rates than the others. This minimal latency is due to the reduction of the number of routers. DRAFT offers also a higher throughput when considering the data rates. This demonstrates a better use of the routing resources than for mesh and fat-tree topologies.

6.3. Scalability. From previous implementation results, the lower resource consumption of the DRAFT network was pointed out. However, it is interesting to verify the scalability of the three topologies considering the network performances and the hardware needs. For this comparison, the networks are implemented with the same characteristics as in previous parts. For the network performances, the highest acceptable data rates are chosen in order to place the networks in the worst conditions. So, the data rate is fixed to 800 Mbit/s for each router. Simulations are realized sending 1562 packets of 16 flits each for an injection time of 1ms. Network performances are presented in Figure 14.

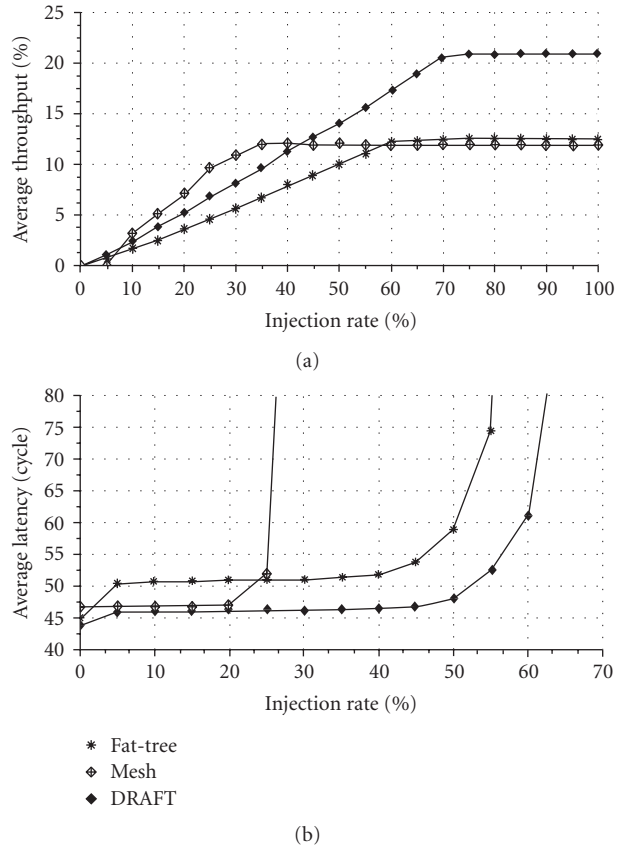

Figure 12: Comparison of the average throughputs (a) and latencies (b) depending on the injection rates for 8 connected CEs.

Thanks to its higher resource consumption, a fat-tree is able to support a higher number of simultaneously connected CEs than other topologies. In these worst transfer conditions, a mesh can manage 9 CEs, DRAFT can handle 10, and the fat-tree supports 13 CEs. Consequently, until 10 simultaneously connected CEs, DRAFT needs less hardware resources and provides a lower latency than other networks. For the connection of more than 10 CEs, the data rates should be restricted or the networks should be adapted for higher network performances at the cost of hardware resources.

The adaptation of the network parameters is now considered in order to improve the network performances. Two parameters are investigated: the flit width and the buffer depth. Results are presented for each parameter considering the required registers and LUTs also with the average latencies. The impact of the flit width is shown Figure 15. The networks are always designed for the connection of 8 CEs, with a buffer depth of 4 flits. The data rate could not be fixed to 800 Mbit/s due to a saturation of the latencies with a flit size of 16 bits. Thus, presented latencies were obtained for a data rate of 400 Mbit/s per CE.

So, the flit size has a great influence over the resource consumption. The impact of this parameter, between 32 and 64 bits, is limited with a data rate of 400 Mbit/s. However, at 800 Mbit/s, it reduces the latencies from 48.05 to 32.31

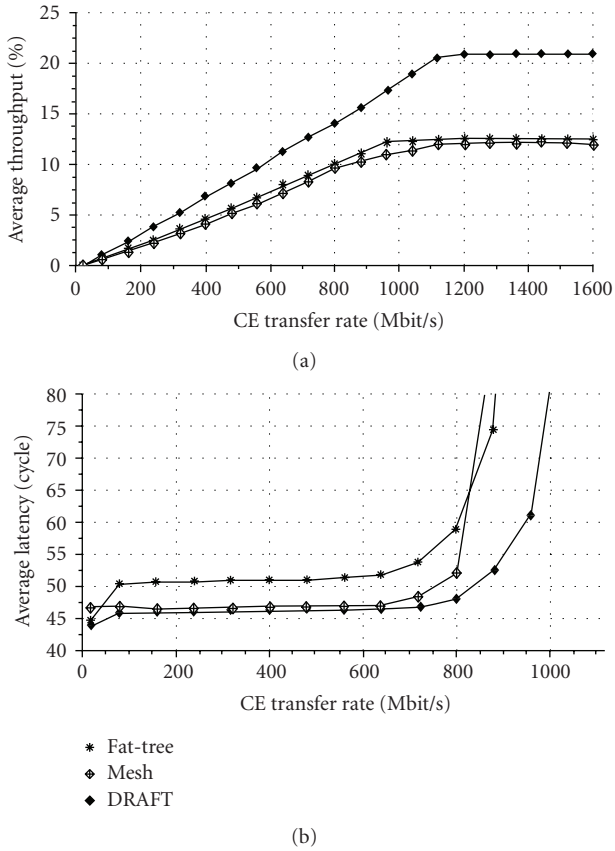

Figure 13: Comparison of the average throughputs (a) and latencies (b) depending on each CE transfer rate.

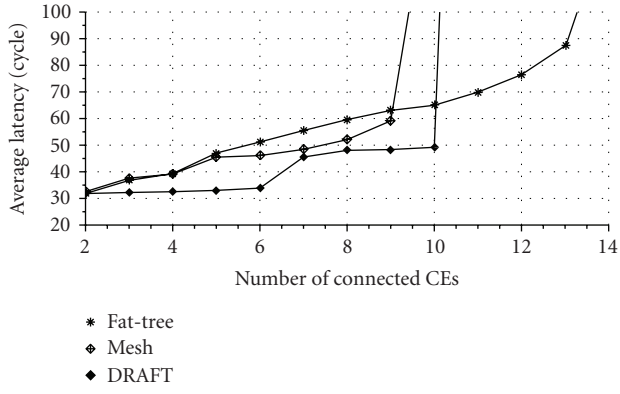

Figure 14: Comparison of the average latencies depending on the number of simultaneously connected CEs.

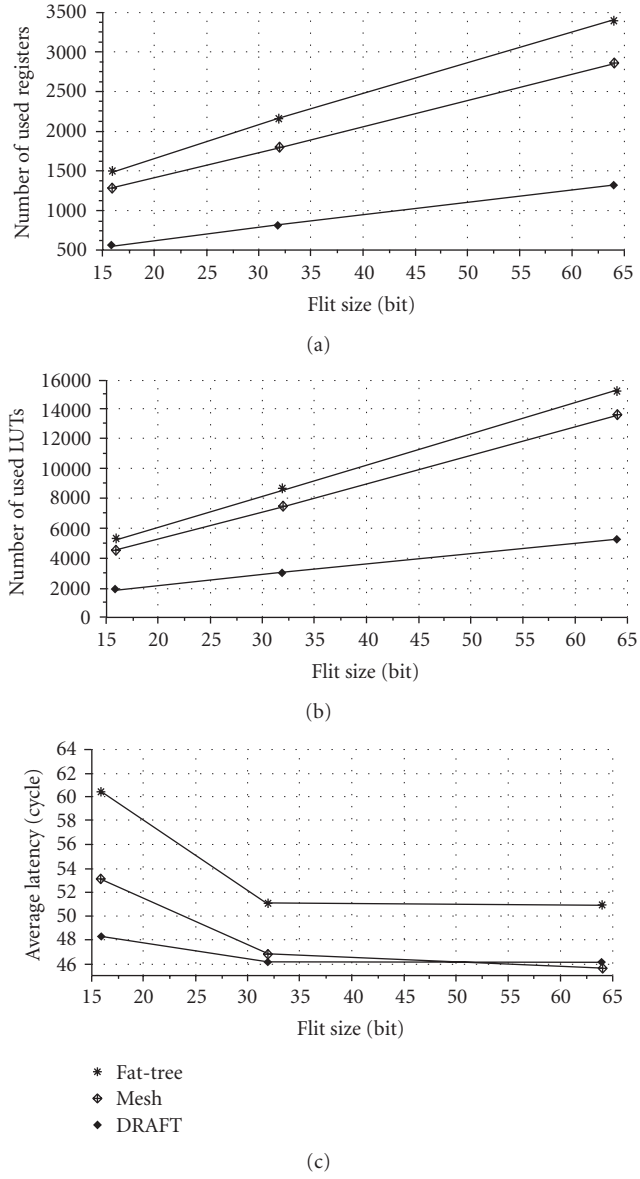

Figure 15: Influence of the flit sizes over the hardware resources (registers (a) and LUTs (b)) and the latencies (c) with a data rate of 400 Mbit/s per CE.

average cycles, respectively, for 32 and 64 bits in DRAFT. This phenomenon can also be observed for the fat-tree and the mesh. Considering the hardware cost of the flit size, this parameter should be minimized according to the desired performances.

Similarly, the impact of the buffer depth is presented in Figure 16. The presented results were obtained with a flit width of 32 bits and with a data rate of 800 Mbit/s per CE.

Concerning the buffer depth, an increase of the depths decreases the average latencies, but its impact over the hardware resources is lower than for the flit size. So, if the flit size is set to a minimum, the buffer depth can be increased in order to reach the network performances and the scalability required by the application.

In conclusion over the scalability, every topology can support a relatively low number of simultaneously connected CEs at full network performances. In order to increase this number, the designer should reduce first the data rate of its CEs. This point is particularly true with the DRAFT topology. If it is not possible, the designer should try to increase the flit size and buffer depth parameters. However, these solutions have an important impact on resource

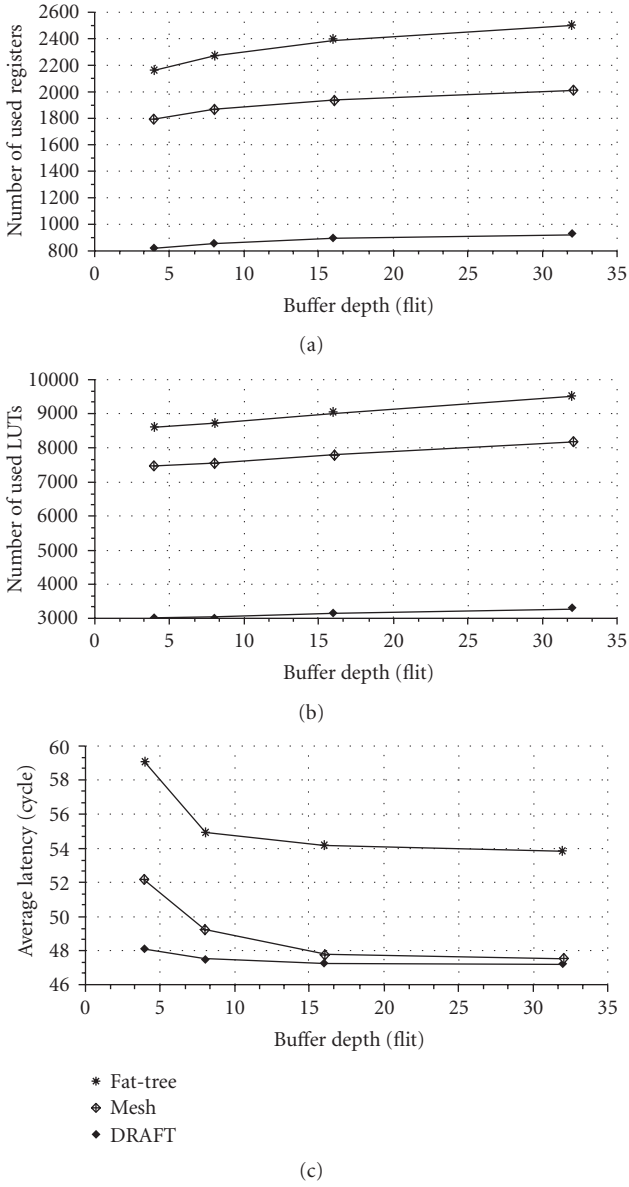

Figure 16: Influence of the buffer depths over the hardware resources (registers (a) and LUTs (b)) and the latencies (c) with a data rate of 800 Mbit/s per CE.

overhead. The use of virtual channels can also increase the scalability of the networks, but the impact over hardware resources is impractical for an implementation in an FPGA.

6.4. Type of Data Traffic. In this part, the influence of the data traffic is presented. Each network is designed to connect 8 CEs with a flit size of 32 bits and a buffer depth of 4 flits. Three types of traffic are studied. Thus, data are produced with a uniform, a normal, or a "pareto on/off" distribution. This latter is a periodic distribution where a period without any emission of data is followed by a period of emission. Results are presented in Table 2.

From these results, DRAFT appears to better support the different types of traffic, even at maximum data rate.

Table 2: Influence of the different data traffics over the latency (average clock cycles).

Type of traffic	DRAFT	Fat-tree	Mesh
Uniform (800 Mbit/s)	48.27	58.22	51.91
Normal (800 Mbit/s)	43.40	56.71	49.21
Pareto (800 Mbit/s)	33.11	42.42	38.40
Uniform (400 Mbit/s)	46.20	51.07	46.87
Normal (400 Mbit/s)	35.58	43.28	38.65
Pareto (400 Mbit/s)	31.43	39.83	35.64

A uniform traffic can be encountered in many applications using an constant flow of data like video processing. The normal repartition of the data rates during computation time is required by applications which depend on the received data like the automatic recognition of a target. The "pareto on/off" corresponds to the traffic between a hardware element and a shared memory using the burst mode, which is the continuous emission of several data during a short period of time. Thus, DRAFT can support all these applications with better performances than other networks.

7. Implementation of an Application Using DRAFT

In this section, the implementation results (hardware resources and network performances) of an application, designed into the framework of the FOSFOR project, are presented. This application is implemented into a middle size Xilinx Virtex5 FPGA: the XC5VSX50T.

A system using DRAFT to interconnect 4 PRRs and 4 shared IOs is implemented with a MicroBlaze processor to control the DPR [31]. The application realizes a target tracking in a video stream. For this purpose, the application is composed of statically implemented tasks, which transform the video stream, and of dynamically implemented ones, which realize the tracking. Dynamically implemented tasks depend on the dynamic number of targets and on the nature of these targets. Thus, these tasks are very heterogeneous in terms of hardware resource requirements. The four shared IOs are located in the central column of the FPGA. This system operates at 100 MHz with a 32 bits data width. In this Virtex 5, DRAFT needs only 2% of the registers and 10% of the LUTs. So, while including the MicroBlaze processor with its memory and peripherals for the DPR management, 92% of the registers and 85% of the LUTs (also with 88% of the BRAMs) remain free for task implementation. Complete implementation results are shown Table 3.

Concerning network performances, the routers are implemented with a critical path of 9,09 ns (110 MHz). DRAFT presents an average latency of 46 clock cycles. In this implementation without virtual channels, it also offers an aggregative bandwidth of 880 MByte/s. A view of the hardware implementation of the system is presented in Figure 17.

Table 3: Complete implementation results of DRAFT interconnecting 4 PRRs and 4 shared IOs.

	Draft alone			Global system		Free space
	Used	Total	%	Used	%	%
Registers	857	32640	2%	2223	6%	92%
LUT	3470	32640	10%	5150	15%	85%
LUT FLIP FLOP	3497	32640	10%	5560	17%	83%
DSP48E	0	288	0%	3	1%	99%
BRAM36	0	132	0%	16	12%	88%

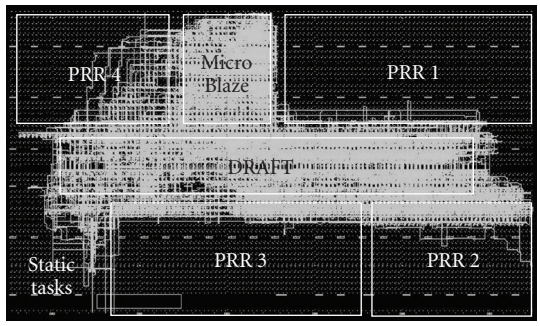

Figure 17: 90-degree rotated view of DRAFT connecting 4 PRRs and 4 shared IOs.

8. Conclusion

In this article, a flexible interconnection network is described. This network is compliant with applications requiring the DPR, and with current FPGA technologies. Thus, from the comparison of current interconnections, even bus-based or NoC based, the fat-tree appeared as a particularly well suited topology for the compliance with the DPR paradigm. Indeed, this structure offers higher network performances than other topologies in terms of bandwidth and latency. A fat-tree is an indirect network that provides high flexibility at the data path level, and supports the parallelization of the communications. Its structure allows interconnecting heterogeneous CEs in heterogeneous FPGAs. The main drawback of this topology is the hardware resource requirements. Hence, the DRAFT flexible network is proposed. DRAFT is indeed the sum of several concepts concerning the structure and the implementation of a fat-tree. These concepts are proposed in order to significantly reduce the resource consumption, and to obtain a network fully compliant with the DPR paradigm. The main idea of DRAFT consists in connecting half of the CEs directly to the root of a fat-tree. The connection of the static elements and the unshared resources is also presented in order to reduce the number of routers, and so the resource consumption. Then, the way to implement DRAFT as a central column into an FPGA is proposed for taking advantage of current FPGA structures.

The DRAGOON generator is designed to parameterize and to automatically generate the DRAFT topology. DRAGOON also supports the simulation of the network allowing to characterize its performances. According with these concepts and thanks to DRAGOON, the DRAFT viability is demonstrated by the comparison with a fat-tree and a mesh network. DRAFT needs fewer resources and fewer communication links than a mesh and a fat-tree. DRAFT presents a lower average latency than every other topology, and supports higher transfer and injection rates (until 1000 Mbit/s). DRAFT is also less sensitive to the flit sizes and the buffer depths than the others so that it can be implemented minimizing its hardware requirements according with the application. Consequently, DRAFT is very well adapted for an implementation into the framework of applications using DPR where there are around 10 simultaneously connected CEs. Finally, the DRAFT viability in terms of compliance with current applications using DPR, and with current technologies, is demonstrated by the implementation of a target tracking application in a Xilinx Virtex5.

Acknowledgment

This research was supported by the ANR (French National Research Agency) within the framework of the FOSFOR project (Flexible OS FOr Reconfigurable devices), http://www.polytech.unice.fr/~fmuller/fosfor/.

References

[1] FOSFOR, http://users.polytech.unice.fr/~fmuller/fosfor/.

[2] D. Cozzi, C. Far, A. Meroni, V. Rana, M. D. Santambrogio, and D. Sciuto, "Reconfigurable NoC design flow for multiple applications run-time mapping on FPGA devices," in *Proceedings of the 19th ACM Great Lakes Symposium on VLSI (GLSVLSI '09)*, pp. 421–424, Boston, Mass, USA, May 2009.

[3] Altera, "Stratix IV Device Handbook—Volume 1," ver 4.0, November 2009.

[4] Altera, "Stratix IV Device Handbook—Volume 2," ver 4.0, November 2009.

[5] ATMEL, "AT40K05/10/20/40AL. 5K–50K Gate FPGA with DSP Optimized Core Cell and Distributed FreeRam, Enhanced Performance Improvement and Bi-directional I/Os (3.3 V)," revision F, 2006.

[6] Xilinx, "Virtex-5 FPGA Configuration User Guide," v3.5, 2008.

[7] "PlanAhead User Guide—version 1.1," Xilinx, 2008.

[8] Xilinx, "Difference-Based Partial Reconfiguration, Application Note XAPP290," 2007.

[9] F. Moraes, N. Calazans, L. Mller, E. Brio, and E. Carvalho, "Dynamic and partial reconfiguration in FPGA SoCs: requirements tools and a case study," in *New Algorithms, Architectures and Applications for Reconfigurable Computing*, pp. 157–168, Springer, New York, NY, USA, 2005.

[10] J. Becker, M. Hubner, G. Hettich, R. Constapel, J. Eisenmann, and J. Luka, "Dynamic and partial FPGA exploitation," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 438–452, 2007.

[11] D. Koch, C. Beckhoff, and J. Teich, "Recobus-builder—a novel tool and technique to build statically and dynamically reconfigurable systems for FPGAs," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '08)*, pp. 119–124, Heidelberg, Germany, September 2008.

[12] A. Thomas and J. Becker, "Dynamic adaptive runtime routing techniques in multigrain reconfigurable hardware architectures," in *Field Programmable Logic and Application*, vol. 3203 of *Lecture Notes in Computer Science*, pp. 115–124, Springer, Berlin, Germany, 2004.

[13] S. Winegarden, "Bus architecture of a system on a chip with user-configurable system logic," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 3, pp. 425–433, 2000.

[14] T. Seceleanu, J. Plosila, and P. Liljeberg, "On-chip segmented bus: a self timed approach," in *Proceedings of the Annual IEEE International ASIC/SOC Conference—System-on-Chip in a Networked World*, pp. 216–220, September 2002.

[15] A. Ahmadinia, C. Bobda, J. Ding, et al., "A practical approach for circuit routing on dynamic reconfigurable devices," in *Proceedings of the International Workshop on Rapid System Prototyping (RSP '05)*, pp. 84–90, Montreal, Canada, June 2005.

[16] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[17] E. Salminen, A. Kulmala, and T. D. Hamalainen, "Survey of network-on-chip proposals," http://www.ocpip.org/white_papers.php.

[18] F. Moraes, N. Calazans, A. Mello, L. Moller, and L. Ost, "Hermes: an infrastructure for low area overhead packet-switching networks on chip," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 69–93, 2004.

[19] C. Bobda and A. Ahmadinia, "Dynamic interconnection of reconfigurable modules on reconfigurable devices," *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 443–451, 2005.

[20] C. Bobda, A. Ahmadinia, M. Majer, J. Teich, S. Fekete, and J. van der Veen, "DyNoC: a dynamic infrastructure for communication in dynamically reconfigurable devices," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '05)*, vol. 2005, pp. 153–158, Tampere, Finland, August 2005.

[21] T. Pionteck, R. Koch, and C. Albrecht, "Applying partial reconfiguration to networks-on-chips," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '06)*, pp. 155–160, Madrid, Spain, August 006.

[22] "Cell Broadband Engine Programming Handbook," IBM, version, 1.11, 2008.

[23] C. Neeb and N. Wehn, "Designing efficient irregular networks for heterogeneous systems-on-chip," *Journal of Systems Architecture*, vol. 54, no. 3-4, pp. 384–396, 2008.

[24] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, 2005.

[25] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, pp. 71–121, 2006.

[26] J. L. Hennessy and D. A. Patterson, "Appendix E: interconnection networks," in *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Mateo, Calif, USA, 2006.

[27] H. Kariniemi and J. Nurmi, "Reusable XGFT interconnect IP for network-on-chip implementations," in *Proceedings of the International Symposium on System-on-Chip*, pp. 95–102, Tampere, Finland, November 2004.

[28] H. Kariniemi, *On-line reconfigurable extended generalized fat tree network-on-chip for multiprocessor system-on-chip circuits*, Ph.D. dissertation, Tampere University of Technology, Tampere, Finland, 2006.

[29] "Synthesis and Simulation Design Guide—ISE 9.2i," Xilinx, 2008.

[30] "ModelSim LE/PE Users Manual 6.5.c," Mentor graphics, 2009.

[31] K. Park and H. Kim, "Remote FPGA Reconfiguration Using MicroBlaze or PowerPC Processors," Application Note: XAPP441 (v1.1) ed., Xilinx.