CuNoC: A Dynamic Scalable Communication Structure for Dynamically Reconfigurable FPGAs

S. Jovanović [a,*], C. Tanougast [a], C. Bobda [b] and S. Weber [a]

[a] *Université Henri Poincaré - Nancy 1*
Laboratoire d'instrumentation et électronique (LIEN)
54506 Vandoeuvre lès Nancy, France

[b] *Department of Computer Science*
University of Kaiserslautern
Gottlieb-Daimler-Str. 48
67653 Kaiserslautern, Germany

Abstract

The growing complexity of integrated circuits imposes to the designers to change and direct the traditional bus-based design concepts towards NoC-based. Networks on chip (NoCs) are emerging as a viable solution to the existing interconnection architectures which are especially characterized by high level of parallelism, high performances and scalability. The already proposed NoC architectures in literature are destined to System-on-chip (SoCs) designs. For a FPGA-based system, in order to take all benefits from this technology, the proposed NoCs are not suitable. In this paper, we present a new paradigm called CuNoC for intercommunication between modules dynamically placed on a chip for the FPGA-based reconfigurable devices. The CuNoC is based on a scalable communication unit characterized by unique architecture, arbitration policy base on the priority-to-the-right rule and modified XY adaptive routing algorithm. The CuNoC is namely adapted and suited to the FPGA-based reconfigurable devices but it can be also adapted with small modifications to all other systems which need an efficient communication medium. We present the basic concept of this communication approach, its main advantages and drawbacks with regards to the other main already proposed NoC approaches and we prove its feasibility on examples through the simulations. Performance evaluation and implementation results are also given.

Key words: Network-on-chip (NoCs), Reconfigurable devices, FPGAs, Dynamic placement of modules, Partial reconfiguration

1 Introduction

Currently, the most frequently used on-chip interconnection architecture is the shared medium arbitrated bus, where all communication devices share the same transmission medium. The major advantages of the shared-bus architecture are simply technology, low area cost and extensibility. On the other hand, this medium allows only one communication transaction at the time [1]. This fact limits the number of the modules (i.e IPs) which can be connected to the bus. Furthermore, the relatively long bus line and the intrinsic parasitic effects mostly caused by additional connected modules increase significantly a propagation delay. This delay may exceed the targeted clock period resulting in an unsuitable functioning of the system.

An alternative to these bus-based interconnection architectures is a network-centric approach [2, 3]. In the network-centric approach often called Network on Chip (NoC), the communication and data transfer between modules (i.e. IPs) can take place in the form of packets. These approaches are also characterized by the explicit parallelism, high modularity and high performances. The communication-centric approaches are especially suitable for high-performance parallel computing. Several papers address this subject and several different interconnection architectures have been proposed [6–13].

In the last decade, the FPGA-based systems emerge as a new computation paradigm. Several processing units (i.e. IPs) can be implemented at a given time and dynamically replaced (entirely or partially) by means of the reconfiguration [14, 15]. The main characteristics of these systems are high level of modularity and flexibility. To design a complex system in the FPGA technology, the system which is built of several processing units, especial attention must be paid on communication medium. The techniques used by now do not take into account all advantages of the FPGA technology, because the system's processing units communicate via bus-based macros. To benefit from all advantages of this reconfigurable technology, a new communication paradigm based on the network-centric approach must be developed. Already presented NoCs are not suitable and appropriated to the FPGA-based systems. The major problem arises when the components need to be dynamically placed on the chip [16].

This paper details the CuNoC, a new interconnection designed for the FPGA-based systems. The basic concept of the CuNoC was firstly presented in [4,5].

* Corresponding author. Tel: +333-8368-4156; fax: +333-8368-4153
Email addresses: slavisa.jovanovic@lien.uhp-nancy.fr (S. Jovanović), camel.tanougast@lien.uhp-nancy.fr (C. Tanougast), bobda@informatik.uni-kl.de (C. Bobda), serge.weber@lien.uhp-nancy.fr (S. Weber).

The CuNoC represents a scalable modular dynamic communication medium whose structure can be reconfigured and adapted depending on the computation needs of the system. In the CuNoC, the reconfigurable resources can be used to implement either the routers in the network or processing elements. The CuNoC is based on a packet-switched network of intelligent routers called *Communication Unit - CU*, hence CuNoC. The main originalities of these routers are novel adaptive routing algorithm and low-overhead implementation compared to the other NoCs. Moreover, the CuNoC presents a very good compromise between used ressources and performances.

This paper is organized as follows. Section 2 gives an overview of related works on the main network-on-chip architectures. Section 3 describes our communication approach and details its basic element, the communication unit (CU). The CU's architecture and the possible uses in several network structures for different communication needs are both presented. The simulation results are given in Section 4. More precisely, some different processing module dispositions are made on the 4x4 CuNoC and simulated. The insertion of a module at run-time is also presented. In Section 5, implementation results on Xilinx Virtex FPGA technology and performance evaluation are presented. Future work and some conclusions are both given in Section 6.

2 Related work

A network, and therefore its complexity is described by two parameters: topology and routing algorithm. According to the topology, the NoCs can be classified in two classes: *static* and *dynamic*. Several static NoCs are presented in literature. Among different static NoC architectures, we distinguish: SPIN [6,7], CLICHÉ [8], Torus [9], folded 2D Torus [10], Octagon [11,12] and BFT [13]. Each architecture is characterized by different topology, communication mechanism, switching mode, routing algorithm, number of modules which may be connected and performances (i.e. transfer data rate) [17,18]. The mostly used network topology and routing algorithm are the 2D Mesh topology and XY routing algorithm.

In these static NoCs, the modules (i.e. processing elements - PEs) are placed in rectangular tiles on the chip and communicate with other modules via the fixed network structure. These fixed networks are composed of intelligent switches which allow communication between all processing elements located in each tile. Each switch associated to a tile interfaces with the network over input and output ports. An output port is used to send packets from the tile whereas an input port delivers packets to the tile. Each switch is characterized by the used *switching technique* or *communication mechanism* [18]. Switching techniques determine when and how internal switches connect their inputs to outputs

and the time at which message components may be transferred along these paths. There are different types of switching techniques: Circuit Switching and Packet Routing [17]. In the circuit switching, a physical path from source to destination is reserved prior to the transmission of the data. The path is held until all the data have been transmitted. In the packet routing, data is divided into fixed-length blocks called packets so, instead of establishing a path before sending any data, whenever the source has a packet to send, it transmits the data. The packet routing requires the use of a switching mode, which defines how packets move through the switches. We distinguish the store-and-forward, virtual cut-through and wormhole switching mode [19]. In store-and-forward mode, the switch cannot forward the packet until it has been completely received. In virtual cut-through mode, the switch can forward the packet as soon as the adjacent switch gives the guarantee that the packet will be accepted completely. In the wormhole mode, the packets are divided into fixed length flow control units (flits) and the input and output buffers are expected to store only a few flits. The first flit, i.e. the header flit, of the packet contains routing information. The header flit decoding enables the switches to establish the path and subsequent flits simply follow this path in a pipelined fashion. As a result, each incoming data flit of a message packet is simply forwarded along the same output channel as the preceding data flit and no packet reordering is required. If a certain flit faces a busy channel, subsequent flits also have to wait at their current locations.

The routing algorithm defines the path which will be taken by the packet between the source and destination. We distinguish the static NoCs with a *static* and *distributed* routing algorithm. In the static routing algorithm, the path is computed at the source before packet sending whereas in the distributed routing algorithm the path computes at the switch level.

A static NoC presents a viable communication infrastructure with many advantages in regard to bus-based platforms. However it is too inflexible for the communication among reconfigurable systems' modules which need a changing network.

The DyNoC was presented in [16], [20] as a medium supporting communication among modules which are dynamically placed on a run-time reconfigurable device. The dynamically placed modules in DyNoC "cover" the routers which were at their place before placement. The covered routers are deactivated and cannot be used for the communication between modules. However, they can be used as modules' additional logics. Once the function of dynamically placed module has finished and the module is removed from the chip, the routers reactivate. This makes the network dynamic. The DyNoC based on S-XY routing algorithm for surrounding obstacles (dynamically placed modules) is presented on mesh topology. According to the authors it can be used for all other topologies [16]. That makes it a logical choice for dynamic changing

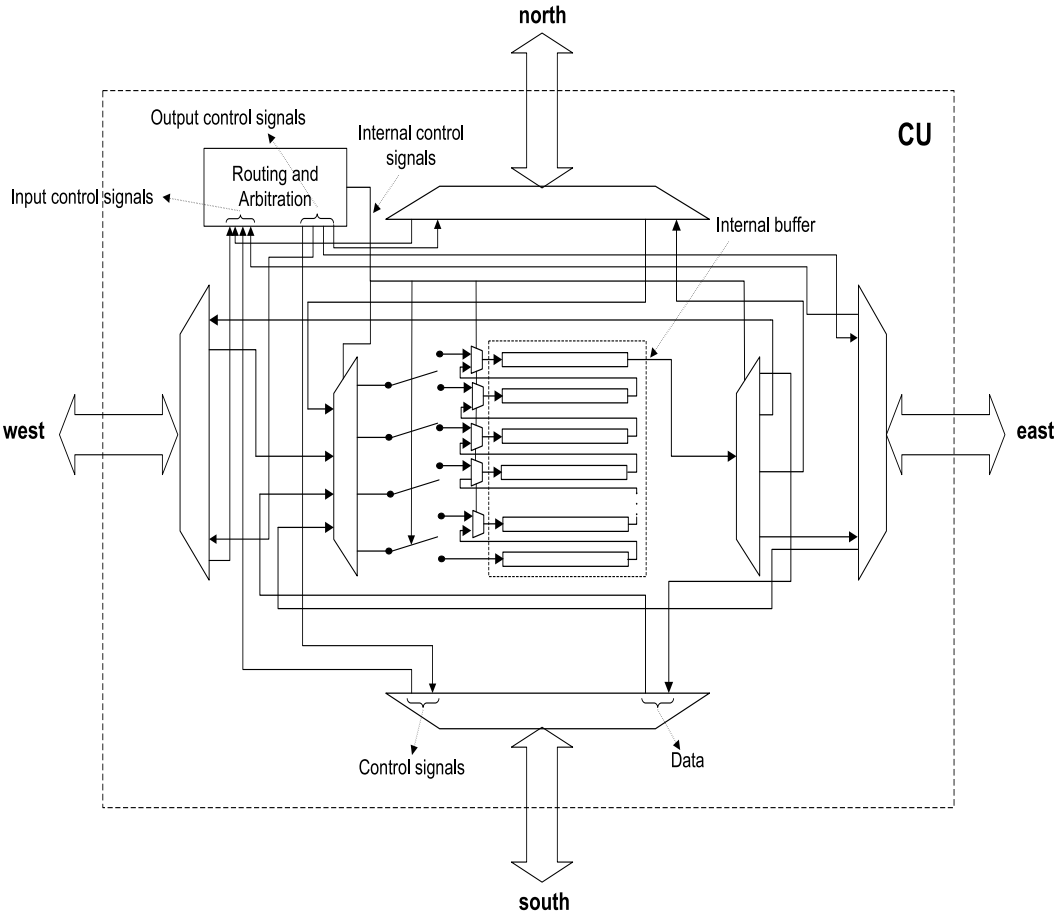

Fig. 1. CU's architecture

networks. On the other hand, for communication among modules in reconfigurable devices, notably on the FPGA-based systems, despite all the advantages, the DyNoC is not very suited communication medium. Significant occupied area by one network element NE (router) [16] and the small area ratio PE/NE (processing/network element) are the main inconveniences for the FPGA-based reconfigurable devices.

3 CuNoC

We present and detail a new NoC-based communication approach called CuNoC for the FPGA-based reconfigurable devices [4,5]. This approach allows communication between modules dynamically placed at the run-time on the chip. The CuNoC represents packet-switched network of intelligent independent routers called *Communication Units* (CUs). Their main role is to route the addressed data packet from the source to its destination in a dynamically changing network. The CUs are characterized by an adaptive routing algorithm, a novel switching policy based on the priority-to-the-right rule, their unique structure

and specific connection with the processing elements (PEs).

3.1 Message structure

Processing elements communicate via data messages. The message is composed of fixed number of packets. Packet's format is depicted in Figure 2 [5]. The first field denotes the destination address whose size depends on a total

Fig. 2. Packet's format

number of the CUs which can be implemented in the reconfigurable area. The second field contains the packet length information, while the third field denotes packet's ID. We consider that the packet's ID size is lower than or equal to the size of the message but greater than zero. Finally, the last field contains data information. Thereafter, by the packet's size we consider the data field's size.

3.2 Communication Unit (CU)

The basic element of our communication approach is the *Communication Unit (CU)*. The main function of the CU is routing of the received packet according to the address contained in the packet. The CU can be either used as a stand-alone element to getting through the messages between up to 4 computing modules or can be used as a part of the network for much more important communication needs. This is the main originality of the CuNoC. The CU's architecture is depicted in Figure 1 [5].

The CU does not have input/output buffers which are used to store temporarily the packets on entries in bottleneck situations, but it has one for all 4 inputs. Consequently, this fact significantly reduces the need for the memory resources on the chip. To manage the routing of several packets which occurred at a given time (up to 4, from 4 directions, Figure 1), the CU uses an arbitration policy based on the rule of the priority to the right. By analogy, we can consider the packets behave like vehicles arriving at one intersection with no traffic lights or other signalization. To get priority, the rule of the priority-to-the-right should be applied.

Once the packets occur at the CU's input ports, the CU "receives" them all at the same time, and treats them, one by one, according to the scheduling imposed by the priority-to-the-right policy. The packet coming from one direction and which had not other packets to its right at the arrival time, has

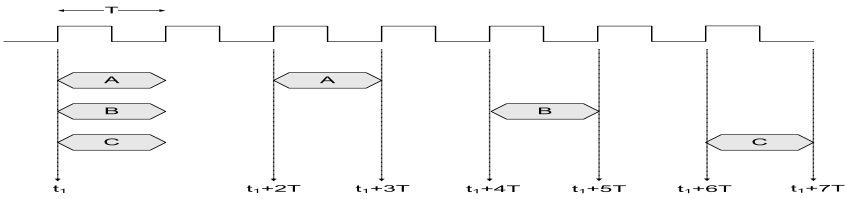

Fig. 3. Illustration of the CU latency on example of three received packets

the highest priority and will be transferred as first. More precisely, the packet with the highest priority is placed in the first register of the buffer and will be treated first, a packet which priority is lower in the next internal register and will be treated second, and so on. In the situation when 4 packets arrive at the same time, the highest priority packet is defined by the designer. The packet treatment means packet routing from one CU to adjacent one until the final destination contained in the destination address field (see Figure 2). The CU uses a *store-and-forward* switching mode. That means the packet cannot be forwarded until it has been completely received. The CU then examines its content and decides where it will be transferred. This fact introduces an additional latency per CU. In our case, the latency of the CU is 2 clock periods per received packet. In example in Figure 3, for 3 packets received at the same time, the packet treatment takes 6 clock periods and after this time, the CU is ready to receive new packets. The procedure of packet treatment is applied to all packets waiting for their transfer to the given directions in the buffer and is described by the following.

In classic XY routing algorithm, the packet is firstly routed in X direction and secondly in Y direction until the final destination [24]. This routing algorithm is not suitable for the dynamically changing networks, because dynamically placed computing module could make maintained communication between two modules worse, or even impossible. In the DyNoC, to make the NoC capable to cope with dynamically placed modules, the authors propose S-XY (surrounding XY) routing algorithm [25]. When a packet reaches an obstacle component placed dynamically, it chooses one of the two possible directions. If the packet moves in the x-direction it will choose to move either upward or downward, whereas if it moves in the y-direction it will choose to move either to the right or to the left. To avoid *ping-pong* game effect which occurs in the situation when one of the actual packet's coordinates is same to the destination's coordinate, the routers stamp the packet to notify their adjacent routers not to send the packet back. Moreover, for all routers it is fixed in advance the direction in which to send a packet when it encounters an obstacle. This can lead to extremely long routing paths.

The CU's routing policy is based on modified adaptive XY routing algorithm. The CU compares its address with the address of the received packet and sets the direction signals (detailed in the next subsection). To transfer the received packet, the CU takes into account the network conditions, that means occupa-

tions of its direct neighbouring CUs. The packet sent via the CuNoC alternates between the horizontal and vertical routing depending on the network traffic. We can say that the packet "searches" for a free and available way to the final destination by crossing the CUs which are on its way. Moreover, the CU takes also into account the direction from which the received packet has arrived to calculate the next packet's direction and to set the direction signals. Thus, the used routing mechanism does not allow the received packet to take the same direction of the arrival. This avoids, among others, the above mentioned ping-pong game effect between the two CUs. The applied routing algorithm sometimes gives longer way to destination than the S-XY algorithm used in DyNoC communication approach. This is due to the fact that the CuNoC's routing algorithm takes at the same time into account the network traffic and dynamically placed modules. That means, between source and destination, in the case where there are not modules placed dynamically or statically, the way taken by a packet is not always straight-forward as it is presented in S-XY algorithm. This aspect can be seen in more details in Section 4.

The major difference between the CU and the other presented routers is that the CU does not have an input/output port for a computing module. It can be either connected via one or several (up to 4) input/output ports to another CU or connected to the computing module (PE). This aspect is more detailed in Section 3.5. In the routing phase where the CU computes the next direction of the received packet, an analysis of all signals of its direct neighbouring CUs is also done. This analysis allows the CU to carry out precisely next packet's direction and to prevent wrong address situations. For example, even packet direction signals, network conditions and direct neighbours occupation indicate a direction to take out, a packet will not take out that direction if it is about a computing module whose address is not contained in the packet destination field.

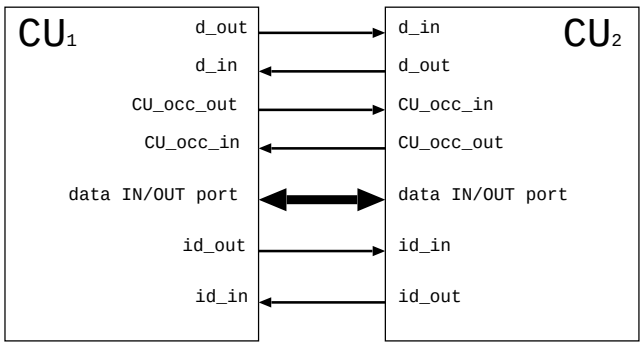

Fig. 4. Physical interface between two CUs

In Figure 4 is presented physical port interface between two CUs. This interface consists of a certain number of input and output signals and of bidirectional data port. Each CU at each port (North - N, South - S, Est - E and West - W) has control and data signals. When the CU sends a packet to its neighbouring CU it sets the control signal *d_out* to 1. That way the neighbouring CU is aware of packet sending. The control signal *d_out* is never set to 1 if the neighbouring CU is occupied. The CU's control logic takes care about all neighbouring CUs' occupancies and generates output control signals at each CU's port. If the *CU_occ_in* is set to 0 at one CU's port, that means the neighbouring CU which is connected to that port is not occupied and CU can send the packet to it. On the other hand, through the *CU_occ_out* control signal the CU indicates to its neighbouring CUs its occupation state.

The *d_in* control signal indicates that the CU will receive a packet. In that case, the CU arbitration logic decides (if there are other packets at other ports) which register of the internal buffer the received packet will occupy and its transfer order. The CU's control logic takes care that the CU never receives a packet if its state is on "occupied". That way, the packets never could be lost in the CuNoC.

As is presented in Figure 4, in the CU's physical interface it can be seen the control signals *id_in* and *id_out*. As we mentioned in Section 3.2, the CU can be connected either to another CU or to processing element. In order to distinguish its neighbours and to inform them about its "nature", the CU uses the *id_out* and *id_in* control signals. In the case of the CU, the *id* signal is always set to 0, otherwise it is set to 1.

For data exchange, the CU at each port has a bidirectional input/output port. That means, two neighbouring CUs cannot send the packets at the same time to each other. The data bus between CUs is shared. In Figure 5 is presented the case of establishing the 4 communication connections through one CU. It

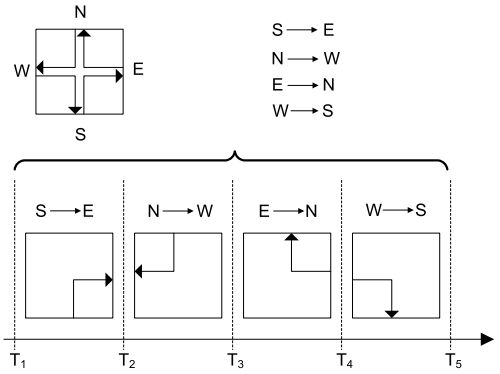

Fig. 5. Time Multiplexed connection between modules in the CuNoC

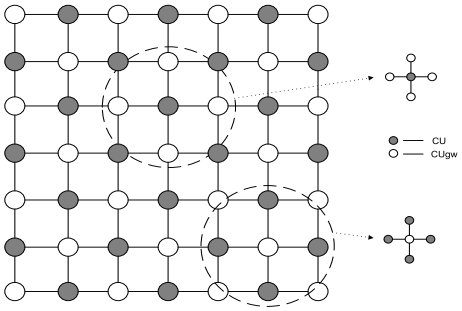

Fig. 6. Valid CUs' position in the CuNoC

can be seen that for each processing elements pair waits its turn to establish a connection and send a packet. The order in which the processing elements establish the connections is defined by the used arbitration policy based on the priority-to-the-right rule.

3.4 Types of CU

Figure 6 presents a regular structure of the CUs in the CuNoC. We distinguish two types of CU: the classic CU and to-give-away CU (CUgw). All CUs behave in the same way in normal circumstances, which means in situations where bottleneck does not occur. The control flow graphs (CFGs) of the packet treatment for the classic and to-give-way CU are depicted in Figure 7 and 8 respectively [4, 5]. The following scenario is applied:

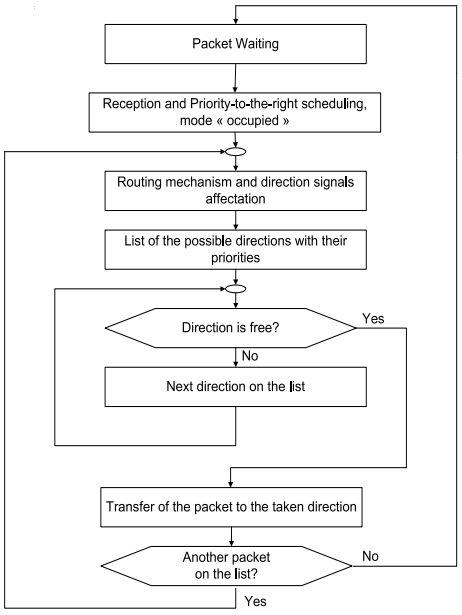

Fig. 7. Control flow graph (CFG) of the classic CU

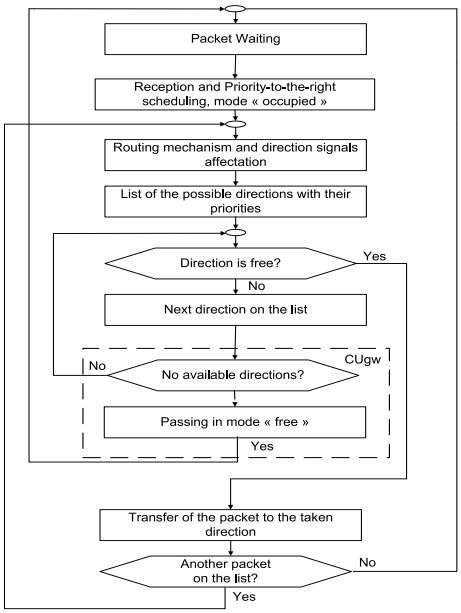

Fig. 8. Control flow graph (CFG) of the CUgw

1. The CU receives all packets (up to 4) at the given time and puts its status on "occupied",

2. According to the priority-to-the-right policy, the CU determines the transfer schedule of the received packets.

3. The CU examines the destination address field and compares it with its address and affects the direction signals: *west*, *east*, *south*, *north*, *stay-x* or *stay-y*.

4. Then the CU tests its direct neighbouring CUs (adjacent nodes), especially those being marked by the direction signals (direction signals put on logic "1"), one by one following the order defined by the priority-to-the-right policy. If the adjacent CU's access is not occupied, the CU transfers the packet to its direction. The CU carries out the same procedure and tests other free accesses until the packet is delivered. If all routing possibilities are exhausted, the bottleneck situation occurres. This means, the situation when all adjacent CUs are in the mode "occupied". In this case, the to-give-way CU (CUgw) passes in action. The concerned CUgw puts on the status "free" and receives all packets of the adjacent CUs. The block in dashed line labeled CUgw in Figure 8 describes this procedure. Once the packets received from the adjacent classic CUs, the CUgw can transfer its packets to the adjacent classic CUs according to their destination address field. On that way, the bottleneck situation is solved. All phases of solving a bottleneck situation are illustrated in Figure 9 in the case of two packets trying to pass trough the network from one to other side.

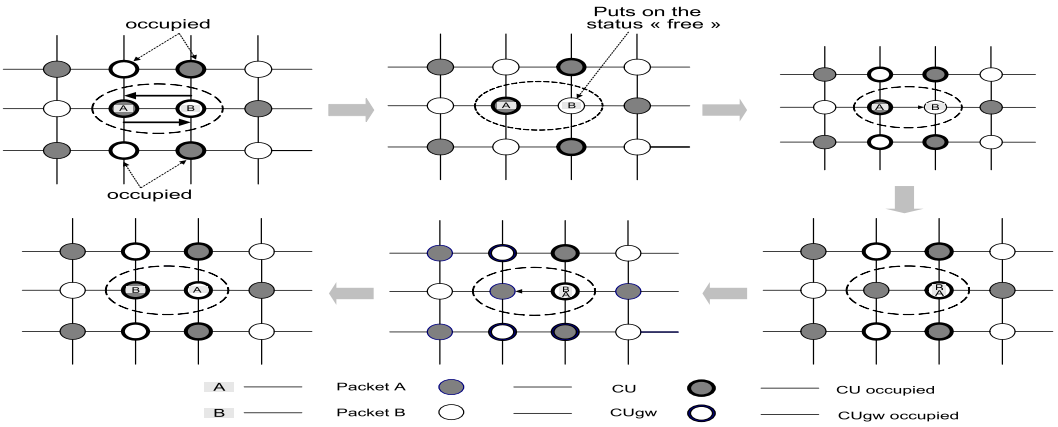

Fig. 9. Example of a bottleneck situation in the CuNoC

5. The CU carries out the same tasks, from step 3 to the end of CFG for all packets being received and scheduled in step 1.

6. Once all packets are transferred to the neighbouring CUs or to the modules, the CU puts on the status "free" and waits for the other packets to be routed. The presented complementarity of the CUs imposes that each classic CU if it would be connected to another CU must be connected and surrounded only by the to-give-way CUs and vice verse.

3.5 Placement and conditions of placement of modules in mesh CuNoC

The CuNoC is especially suited for dynamic placement of modules because the path between the source and destination calculates at run-time. In fact, for different packets of one message, the path between the source and destination is not always the same (except in the case of unilateral communication between only two connected modules). The network situation evolves at the time. Each packet manages to reach its final destination. The CU which has the status "free" becomes "occupied" because it receives some packets, another CU which has the status "occupied" becomes "free" and so on. Thus, by placing a module in the middle of the network between two modules having important intercommunication, their communication will not deteriorate. The packet will consider the dynamically placed module as a temporary obstacle and will try to find another way to its destination getting round the placed module. However, a certain number of rules must be defined and respected in order to ensure an efficient communication between all modules placed statically at compile time or dynamically at run-time on the chip [5]:

Rule 1. Each module (i.e. processing element - PE) has the access to the network by at least one port of the CU which surrounds it.

Rule 2. All PEs communicate via the routers (CU), even the adjacent PEs.

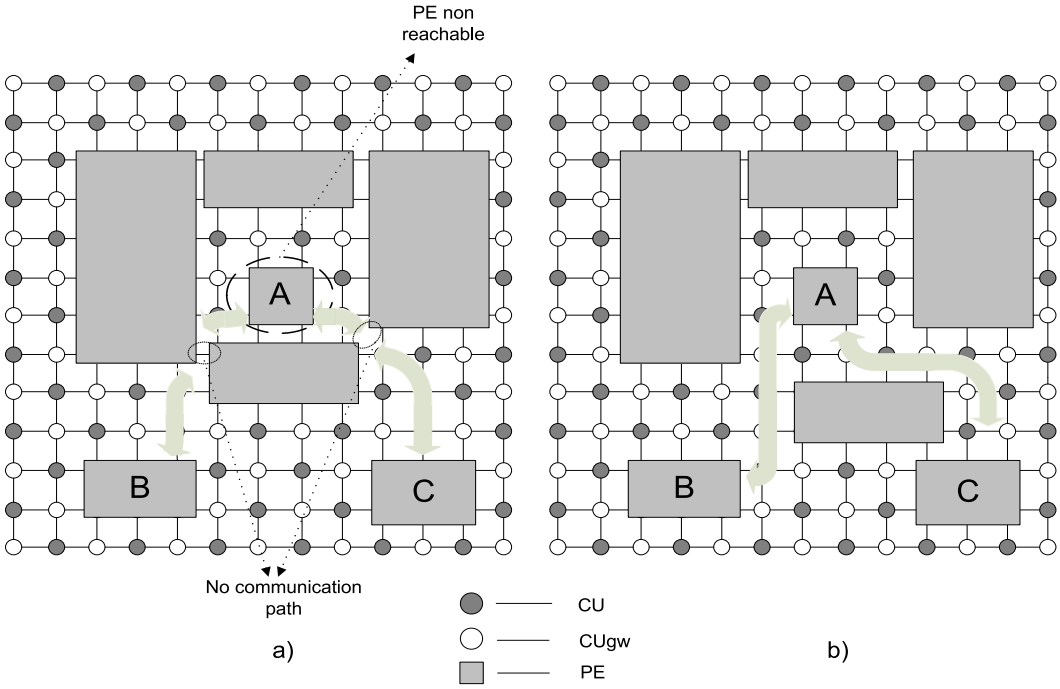

Fig. 10. CuNoC placement: a) non valid b) valid

Rule 3. Network elements (CUs) are either connected to the adjacent CUs or directly to the modules via the CU's ports. Each CU is connected at least to one CU of different type (the classic CU to the CUgw and vice verse, see figure 6).

Rule 4. Each module PE is surrounded by other modules on maximum three sides .

Rule 5. The DyNoC rule: All modules dynamically or statically placed which are surrounded on all sides by the CUs are always reachable [16].

Rule 6. Between all modules there should be at least one path allowing their intercommunication.

In Figure 10 some possible placements of modules at run-time are presented [5]. Figure 10a presents a non valid placement of modules because the above mentioned rule 6 is not respected. The module A placed in the middle of the CuNoC is unreachable because it is surrounded on all sides by other modules. This placement does not allow intercommunication between this module and the modules B and C placed respectively at the bottom left and at the bottom right part of the network. On the other hand, Figure 10b presents a valid placement of modules in the CuNoC. This valid placement is just one among several solutions which takes into account all rules presented above. The all placement constraints presented for the DyNoC in [16, 25] fit well with the CuNoC network. On the other hand, the placement constraints for the CuNoC are less constraining than those for the DyNoC. For example, for the

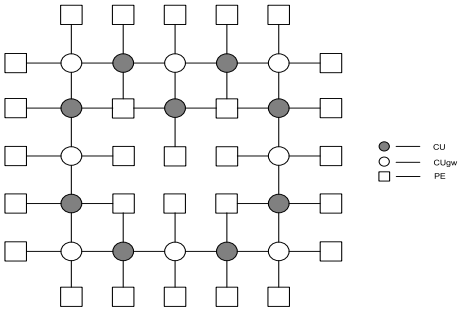

Fig. 11. A possible placement of a maximal number of modules that could be connected in m x n CuNoC

CuNoC, it should have at least one path between all modules (Rule 6) but all modules do not have to be surrounded on all sides by the network elements (CUs).

As we mentioned, the CU has 4 input/output ports and it does not have a connecting access point for the computing module. This fact allows mixing of the CUs and modules in the network. The CUs are well suited for the 2D mesh topology. In Figures 11 and 12 are presented some examples of the CuNoC structures based on the 2D mesh topology [5]. Figure 11 presents a structure of maximal number of modules which can communicate via a m x n 2D mesh CuNoC. The maximal number of modules which could be connected to this CuNoC and which could intercommunicate while respecting above presented rules, is calculated for the minimal module's area size. The minimal module's area size is equal to CU's area size. The maximal number of modules of minimal size (size of one CU) which can be connected to 2D m x n mesh CuNoC is described with the following equation [5]:

$$N_{mod_{max}} = 3m + 4n - 8 + trunc\left[\frac{(n-4)(m-3)}{2}\right] \tag{1a}$$

$$N_{CU} = m * n - N_{mod_{max}} \tag{1b}$$

where N_{CU} is the number of CUs left from initial m x n network. The term $trunc\left[\frac{(n-4)(m-3)}{2}\right]$ is equal to 0 for n and $m \leq 3$.

The structure of modules and CUs presented in Figure 11 is feasible but in practice it would not work well if the computing modules which are connected to the network have important communication needs. In order to increase communication performances, a structure in which each module (PE) has at least one CU connected only to it is more suited (see Figure 12). The other ports of the concerned CU can be connected to the other CUs, but could not be connected to another module. In that case, the maximal number of modules which can be connected to the initial m x n CuNoC is determined by the following equation:

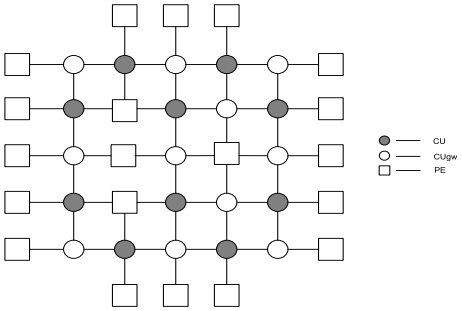

Fig. 12. Preferable placement of the modules in the CuNoC: each module has at least one CU connected only to it

$$N_{mod_{max}} = 2(m+n-2) + trunc\left[\frac{(m-2)(n-2)}{2}\right] \tag{2a}$$

$$N_{CU} = m * n - N_{mod_{max}} \tag{2b}$$

where the term $trunc\left[\frac{(m-2)(n-2)}{2}\right]$ is equal to 0 for n and $m \leq 2$.

3.6 Dynamic placement of modules in mesh CuNoC

The starting point of our approach is the valid placement of CUs on the reconfigurable area. Figure 13 illustrates 3 successive phases of building a mesh CuNoC and placement of modules on it [5]. After having placed the network of routers, the next step consists in the placement of the modules. Each module having the area size lower than the CU's size replaces one CU. For all other modules having the area size greater than the CU's size, we replace a zone of several CUs having total area size equal or greater than the module's area size. The module which covers at the time of the placement one or several CUs inherits all their addresses. That means, the module which

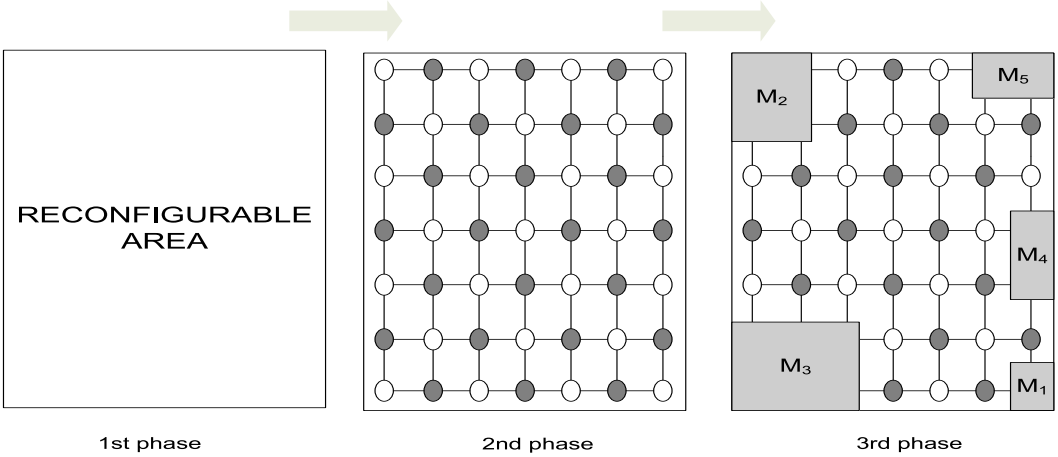

Fig. 13. The first three phases of building a mesh CuNoC

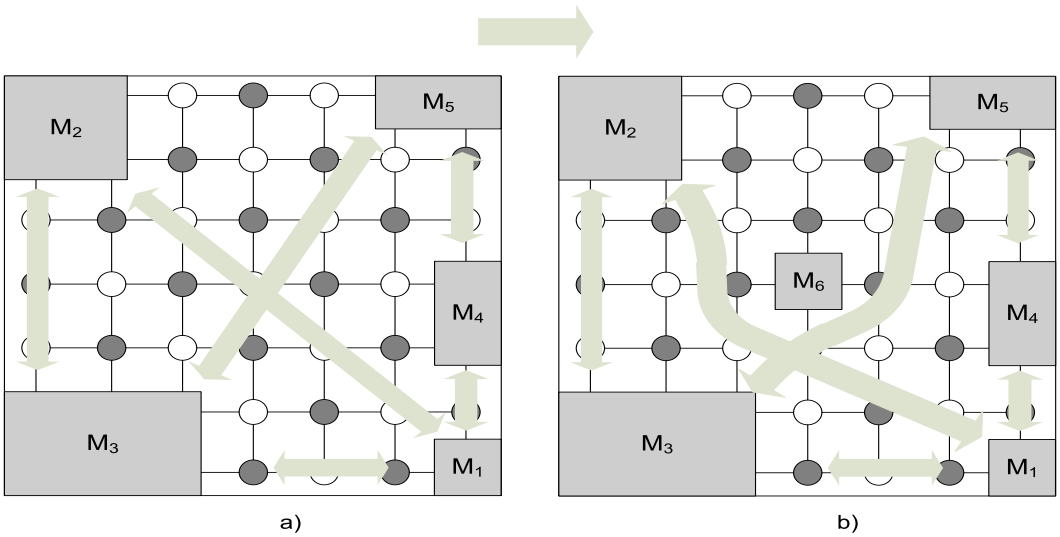

Fig. 14. Dynamic placement of modules in CuNoC

covered 4 CUs at the placement time has 4 addresses and more than 4 access points to the network.

The CuNoC approach is very suitable for dynamic placement of computing modules at run-time. Indeed, let us suppose that we have the following situation: in the 7x7 CuNoC we have statically placed (or dynamically after having placed statically the 7x7 CuNoC) 5 computing modules. Each computing module carries out a certain function and need some information exchange with the other modules placed on the chip. After a certain time, the important communication is established within this network (see Figure 14a). At a given time, a new function demand occurs or a new computing module must be inserted into the network. A reconfiguration area controller which decides on which place of the network a module will be inserted, carries out an analysis of the available free reconfigurable area (area which is not covered with other modules) and it decides which CUs will be replaced with the new computing element. It "informs" all the CUs concerned about the action that will be carried out in order to allow them to empty their buffers and to change their status on "module". The rest of the network will further consider the CUs marked to replace as an obstacle and could not take the way(s) containing these CUs. This situation is illustrated in Figure 14b.

4 Simulation results

In order to validate functionally the CuNoC communication approach, we have simulated several structures of computing modules on the mesh CuNoC. We used the 4x4 CuNoC network and we made some different dispositions of computing modules. In the first case, we connected 16 modules to the 4x4 CuNoC.

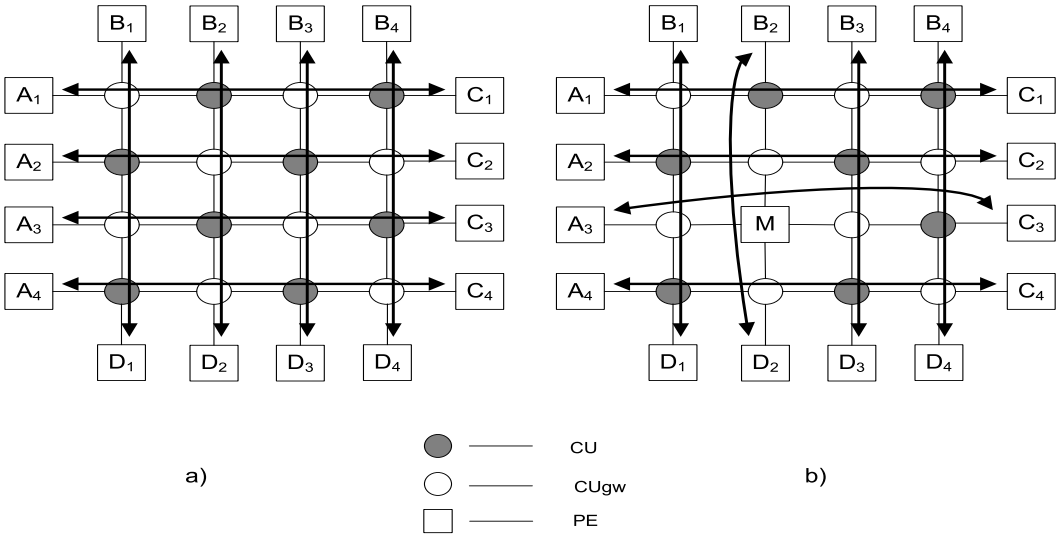

Fig. 15. Simulation structure: a) Communication of 16 modules by 4 x 4 CuNoC b) Processing module insertion

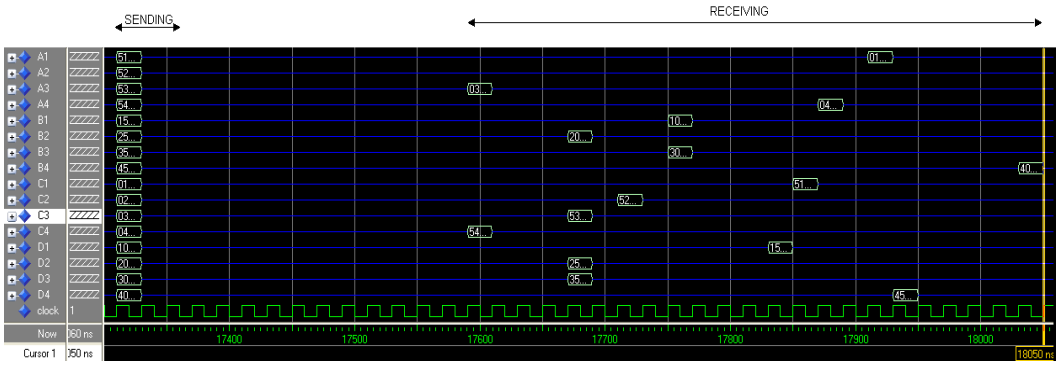

Fig. 16. Simulation results: communication between 16 modules

Each connected module communicates with another one and sends/receives the packets to/from it. These results are presented in Section 4.1. In the second case, we simulated a module insertion in the structure of modules from the previous case. On this example, it can be seen some results concerning the communication data path changing between modules. These results are presented in Section 4.2.

4.1 4x4 CuNoC simulation structure

We simulated a communication between 16 modules via 4 x 4 CuNoC. Figure 15a illustrates the simulated CuNoC topology and disposition of the modules. Figure 16 presents a snapshot of simulation results for this case. It can be seen that all modules send at the same time packets and receive them after the latency period. The impact of the latency on network performances is

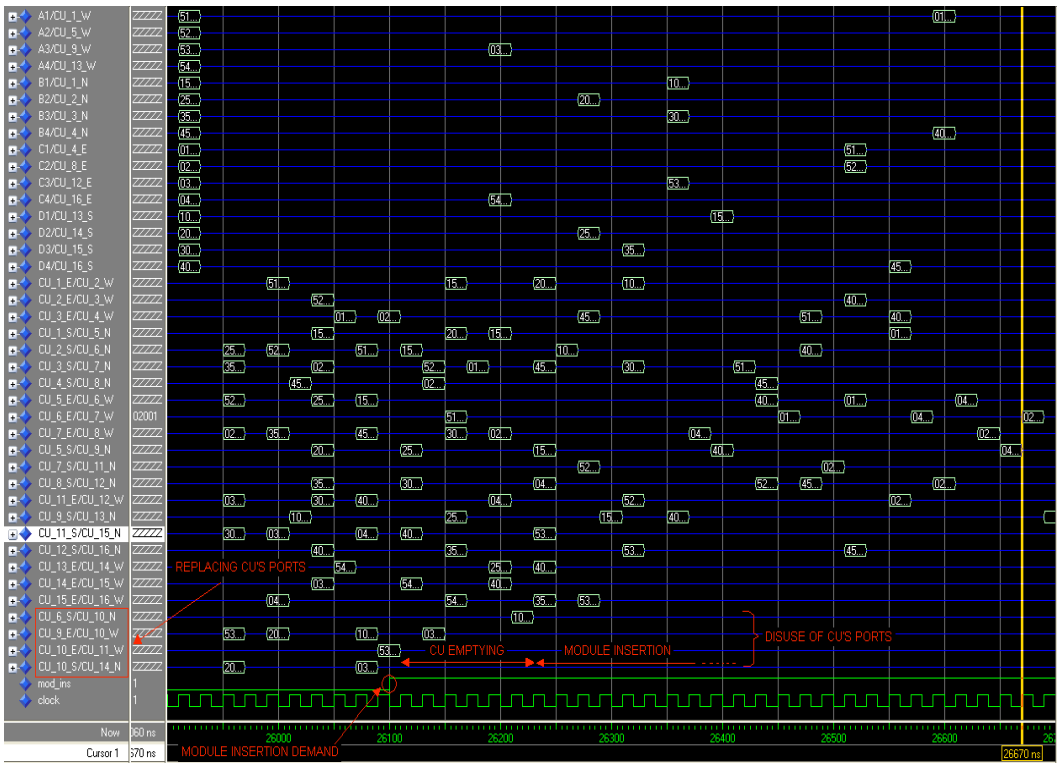

Fig. 17. Simulation results: communication between 16 modules and module insertion

discussed and detailed in Section 5. An inconvenience of this approach is that the certain packets do not arrive in the sending order. If we wait until all modules receive the packets after sending them, there will never be need for packet reordering and packet collision situations will be avoided.

4.2 Processing module insertion

The second simulation case presents a processing module insertion in the structure of modules from the first simulation case, see Figure 15b. The snapshot of this simulation is presented in Figure 17. Firstly, the communication between 16 computing modules is established, as is presented in Figure 16. After a while, the module insertion demand occurs. This demand is specified by the signal *mod_ins* in Figure 17. The CU which will be replaced with the new computing module M (in this case CU_{10}, see Figure 18c) changes its status on "module" and empties its internal buffer. The emptying can take few clock cycles, as it can be seen from simulation snapshot in Figure 17. After having changed its status, the CU is not any more available for neighbouring CUs. That means, the packets from processing modules which pass through this area of the network will furthermore exclude this CU from their communication paths (see Figure 18b).

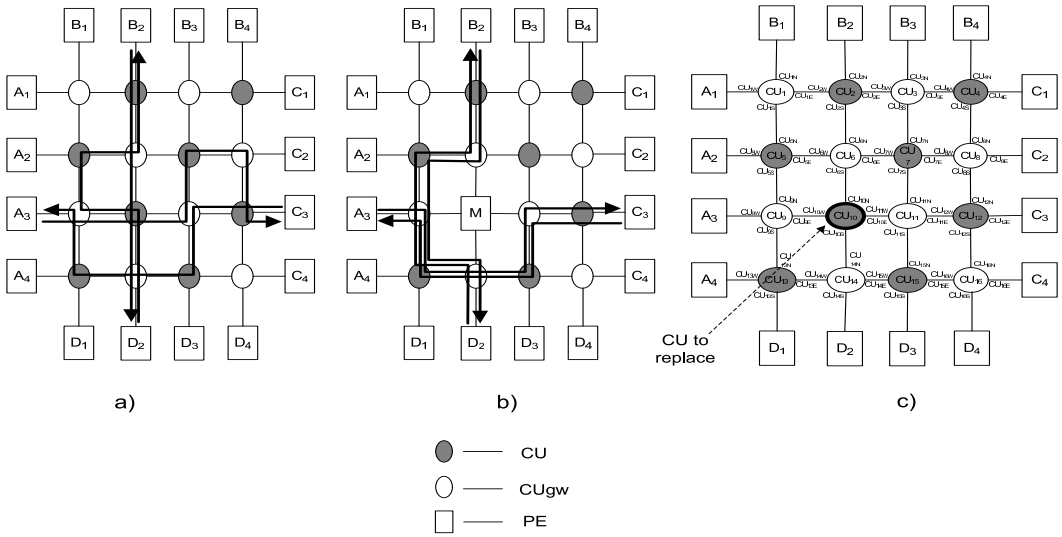

Fig. 18. a) Example of dynamic communication path between 4 modules in the 4x4 CuNoC used for intercommunication among 16 modules before module insertion b) after module insertion c) Processing module M replaces the CU_{10} in the second simulation case, section 4.2

We have analyzed the communication data paths which take the packets from one to other processing modules before and after module insertion. These results are presented in Figure 18. Before module insertion, the communication data path between 2 modules is not necessary straight forward. As it has been stated above, the data path that one packet will take is chosen at run-time. In fact, the packet's path changes progressively in time because the path is not defined at the beginning of packet sending. Moreover, the packet decides dynamically in function of the CUs' occupations and network traffic which path it will take. That explains the communication data paths which were taken by packets before module insertion, Figure 18a. We have observed particularly the communication paths between modules on which way the CU_{10} is located. More precisely, the communications between modules A_3 and C_3 and between B_2 andD_2 are considered. It can be seen that in Figure 18a, before the module insertion, that few packets from these modules choose paths which take the CU_{10} on their way. After module insertion, this is not anymore the case. All packets from the modules take paths which do not comprise the CU_{10} on their way, Figure 18b.

These simulation results show one of the main properties of the CuNoC communication approach - the possibility of dynamic placement of modules at the run-time. The run-time dynamic modules placement will not deteriorate already established communication between the modules. This property can well be used with the partial reconfiguration property of FPGAs which allows the reconfiguration of the FPGA area part [23].

5 Implementations results and performance evaluation

Table 1
CU Statistics

	CU classic			CU tgw		
data width	Virtex II f [MHz]	CLB Slices	Virtex IV f [MHz]	Virtex II f [MHz]	CLB Slices	Virtex IV f [MHz]
8 bit	320.7	49	549.3	302.7	50	549.3
16 bit	320.7	84	549.3	279.6	74	549.3
24 bit	272.1	112	549.3	279.2	98	549.3
32 bit	272.1	140	549.3	279.2	122	549.3
48 bit	272.1	196	549.3	279.2	170	549.3
64 bit	272.1	252	549.3	279.2	218	549.3
128 bit	250.0	476	549.3	250.0	410	549.3
256 bit	279.2	924	549.3	279.2	820	549.3
512 bit	244.3	1820	549.3	250.4	1564	549.3

Table 2
CuNoC Statistics

	CuNoC 2x2			CuNoC 3x3			CuNoC 4x4		
data width	Virtex II f [MHz]	CLB Slices	Virtex IV f [MHz]	Virtex II f [MHz]	CLB Slices	Virtex IV f [MHz]	Virtex II f [MHz]	CLB Slices	Virtex IV f [MHz]
8 bit	259.1	197	549.3	241.7	443	549.3	279.6	788	549.3
16 bit	241.6	293	549.3	279.6	659	549.3	213.0	1172	549.3
24 bit	241.6	418	549.3	228.8	948	549.3	259.4	1672	549.3
32 bit	241.6	522	549.3	228.8	1184	549.3	250.4	2088	549.3
48 bit	228.8	730	549.3	250.4	1656	549.3	279.2	2920	549.3
64 bit	252.9	938	549.3	250.4	2128	549.3	272.1	549.3	549.3

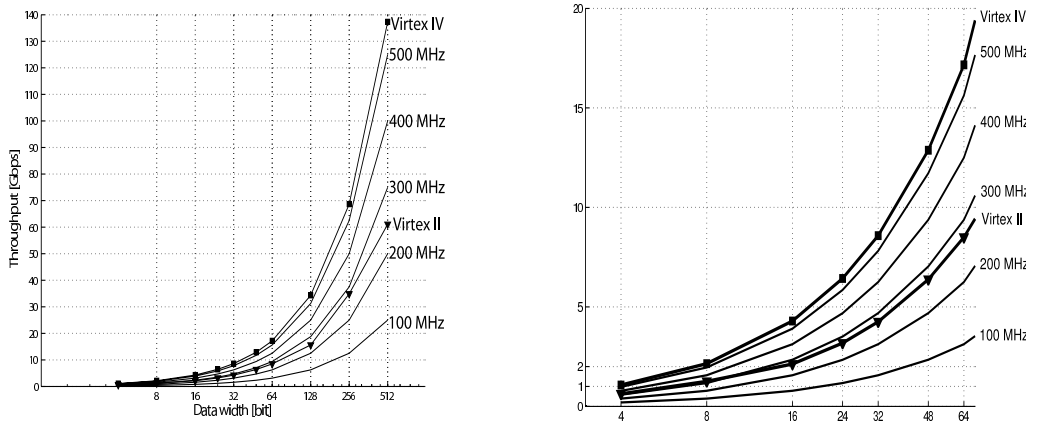

Fig. 19. Throughput of one CU for different data widths: a) data bit width from 4 to 512 bit (left) b) data bit width from 4 to 64 bit (right)

5.1 Implementation results

We have synthesized and implemented the CUs of various data format in Xilinx Virtex II and IV technology. These results are given in Table 1 in CLB slices for area occupation and in MHz for maximum operating frequency f. It can be seen that the 8-bit CU takes only 50 CLB slices and has a maximum operating frequency in Virtex IV technology of 549.3 MHz.

We have also synthesized and implemented different CuNoC sizes in Xilinx Virtex II and IV technology. These results are given in Table 2 also in terms of area occupation in CLB slices for different data widths and maximum operating frequency in MHz.

5.2 Performance evaluation

5.2.1 Throughput

As we mentioned in Section 3.2, the CU uses store-and-forward switching mode which introduces an additional latency per CU. This latency is in range between 2 and 8 clock cycles depending on the number of received packets at the time. The additional latency per switch affects significantly CU's performances.

The maximum throughput of an n data-bit-width CU at frequency f to which the maximum number of processing modules are connected (4 PEs) is given by the following equation:

$$Throughput_{max} = n * f/2 \tag{3}$$

The presented expression is divided by 2 taking into account the maximal latency per CU which is 8 clock cycles for 4 connected modules. For example, for a 8 data-bit-width CU to which 4 processing modules are connected at 100 MHz we have the throughput of 400 Mbps. For other data widths (up to 512) evaluated throughput results are depicted in Figure 19. These results take into account the maximum operating frequencies for different data widths in Virtex II and IV FPGA technology (see Table 1). It can be seen that the for 64 data bit width in Virtex IV technology at maximum operating frequency of 549.3 MHz the maximum throughput for one CU is a bit less than 20 Gbps. Here we must state that the 64-bit CU takes only 252 CLB slices!

One of the reasons, other than the latency per CU, which also leads to decreasing performances, especially in terms of bandwidth, is the time-multiplexed connection between CUs, discussed in 3.3.

In n x n CuNoC, the maximum theoretical throughput cannot be calculated in a classical manner, that means the number of switches n multiplied by the maximum throughput per switch $Troughput_m$. Larger CuNoC allows interconnection among larger number of processing modules, but the maximum throughput does not increase as a multiple of the maximum throughput per CU. The evaluation of bandwidth performances of larger CuNoCs is not considered in this paper.

5.2.2 Latency

In n x n CuNoC the minimal latency of a packet sent from source to target is defined by following equation:

$$latency_{min} = N_{CU} * latency_{CU_{min}} \tag{4}$$

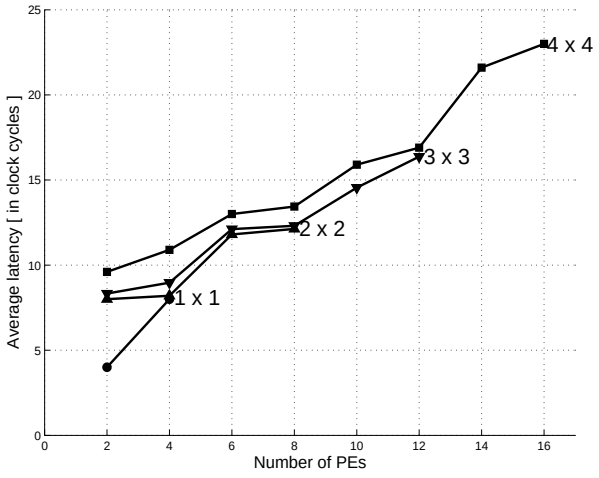

Fig. 20. Average latency in function of the number of connected PEs (in this case traffic generators - TGs)

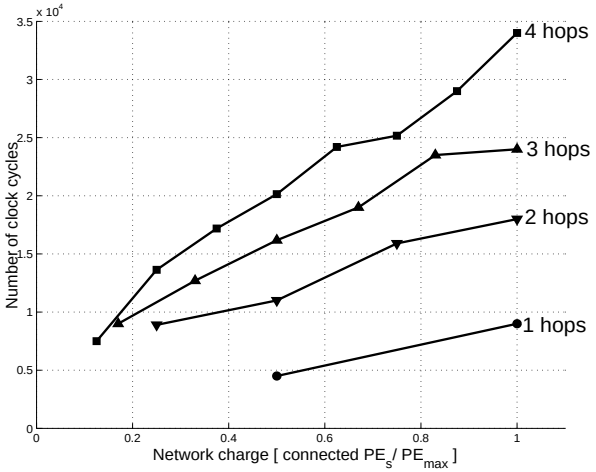

Fig. 21. Number of clock cycles in function of data load being needed to send 1000 packets per connected PE

Table 3
Latency evaluation of 131 072 sent packets per PE with random traffic for 4 different CuNoC sizes

CuNoC	4 x 4	3 x 3	2 x 2	1 x 1
Average	13.5 - 23	9 - 16.4	8 - 13.5	2 - 8
Minimum	8	6	4	2
Maximum	39	25	19	8

Numbers in Tables express clock cycles

where the N_{CU} is the number of switches (CUs) in the communication path, $latency_{CUmin}$ is the CU's minimal latency depending on number of received packets at the time which is in range $2 * T_{CLK} < latency_{CU} < 8 * T_{CLK}$ and T_{CLK} is the clock period.

The latency value varies in function of network traffic. We have evaluated the average latency of different sizes of CuNoC: 1 x 1, 2 x 2, 3 x 3 and 4 x 4. For each topology the number of processing modules connected to has been changing from minimal to maximal (up to 16 for 4x4 CuNoC). As a processing module, we have modelled an VHDL traffic generator. Each processing module (in this case, the traffic generator) communicates with another one placed on the opponent side of the CuNoC. The same topology is used as one presented in Figure 15a for the first simulation case.

In order to avoid blocking of packets, which is possible in the evaluated topology where all processing modules have only one access to the network, we applied the following scenario: at the same time, all traffic generators sent a packet to their paired generators on the other side of the network and wait

the packet sent to them. After having received the packets, all traffic generator repeat the same scenario. The number of packet that the traffic generator sends to its pair is equal to 131 072. Figure 20 presents the average latency in function of the number of traffic generators connected to the network for 4 different sizes. It can be seen that the average latency varies considerably for different CuNoC's sizes. The main explication for this is that run-time establishing of communication path between modules.

Figure 21 presents the result of evaluation of the number of clock cycles needed to send 1000 packets per PE in different data load situations. For example, for a 2 hop distance (source and target not included) between 4 connected PEs (load equal to 0.5), is needed approximately 6 000 clock cycles.

Table 3 presents maximal and minimal latency values for all considered cases. All presented values are in clock cycles. It can be seen that the minimal latency for the 4x4 CuNoC in the case of 16 traffic generators connected to with the distance of 4 CUs is 8 clock cycles whereas the maximal latency is 39 clock cycles.

6 Conclusion and future work

In this paper, we proposed a new dynamic intercommunication structure for run-time reconfigurable modules in FPGAs. We have presented the basic concept of this communication approach, its basic router's architecture, possible topologies and structures mixing the network of CUs with the processing elements (PEs). The CuNoC is functionally validated through the simulation as a module insertion at the run-time. The main advantages and drawbacks as performance evaluation results on for chosen topology are also presented and discussed. The CuNoC performances have been evaluated on mesh topology of different sizes for different dispositions of PEs.

Our communication approach represents an infrastructure which is namely adapted and suited to the FPGA-based reconfigurable devices. The CuNoC represents a scalable network structure which can be used either as a standalone unit for communication between up to 4 PEs or as a part of a huge network. The main advantages of the CuNoC are small area overhead of its basic element(*CU - Communication Unit*) and the possibility of dynamic placement of modules at run-time. From performance evaluation and implementation results we conclude that the CuNoC presents a good compromise between the logic area and operating frequency.

The CuNoC infrastructure provides in its current state support to the implementation of *best effort - BE* services only [26, 27]. That means, there is no

guaranteed quality of service level. For the hard real time applications, *the guaranteed throughput - GT* services must be provided. Another ongoing work consists in employing of one of the interface standards such as the OCP standard [28]. The CuNoC have been tested on mesh topology. Other topologies than mesh will be considered as well as implementation of a given application on the CuNoC.

References

[1] S. Kumar et al.:*A Network on Chip Architecture and Design Methodology*, in IEEE Computer Society Annual Symposium on VLSI (ISVLSI'02), April 2002.

[2] L. Benini and G. De Micheli: *Networks on chips: a new SoC paradigm*, IEEE Computer, jan 2002.

[3] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg and D. Lindqvist: *Network on chip: An Architecture for Billion Transistor Era*, Proc. of the International NorChip Conference, sep 2000.

[4] S. Jovanovic, C. Tanougast, C. Bobda and S. Weber: *CuNoC: A Scalable Dynamic NoC for Dynamically Reconfigurable FPGAs*, Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on, 27-29 Aug. 2007, pages 753-756, 10.1109/FPL.2007.4380761.

[5] S. Jovanovic, C. Tanougast, C. Bobda and S. Weber: *A Dynamic Communication Structure for Dynamically Reconfigurable FPGAs*, Reconfigurable Communication Centric SoCs, ReCoSoC07, June 2007, Montpellier France.

[6] P. Guerrier and A. Greiner: *A Generic Architecture for On-chip Packet-Switched Interconnections*, Proc.Design and Test in Europe (DATE), pages 250-256, mar 2000.

[7] A. Andiahantenaina, A. Grenier: *Micro-network for SoC: implementation of a 32-port SPIN network*, in: Design Automation and Test in Europe (DATE'03), Pages 1128-1129, March 2003.

[8] S. Kumar et al: *A Network on chip Architecture and Design Methodology*, Proc. IEEE Computer Society Annual Symp. on VLSI, pages 117-124, 2002.

[9] T. Marescaux, A. Bartic, D. Verkest, S. Vernalde and R. Lauwereins: *Interconnection Networks Enable Fine-Grain Dynamic Multi-Tasking on FPGAs*, Proc. of 12th International Conference, FPL, Montpellier, France, sep 2002.

[10] W. J. Dally and B. Towles: *Route Packets, Not Wires: On-Chip Interconnection Networks*,Proc. Design Automation Conf. (DAC), pages 683-689, 2001.

[11] F. Karim et al: *An Interconnect Architecture for Networking Systems on Chips*, IEEE Micro, volume 22, number 5, pages 36-45, mar 2002.

[12] F. Karim et al: *On-chip Communication Architecture for OC-768 Network Processors*,in: 38th Design Automation Conference (DAC'01), Pages 678-683, June 2001.

[13] P. P. Pande, C. Grecu, A. Ivanov and R. Saleh: *Design of a Switch for Network on Chip Applications*, Proc. Int. Symp. Circuits and Systems (ISCAS), pages 217-220, volume 5, may 2003.

[14] S. A. Guccione and D. Levi: *The Advantages of Run-time Reconfiguration*, In John Schewel et al editors, Reconfigurable Technology: FPGAs for Computing and Applications, Proc. SPIE 3844, pages 87-92, Bellingham, WA, sep 1999.

[15] P. Lysaght, J. Dunlop: *Dynamic reconfiguration of FPGAs*, in: W. Moore, W. Luk, (Eds), More FPGAs, Proceedings of the International Workshop on Field-programmable Logic and Applications, pages 82-94, 1993.

[16] 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*, International Conf. on FPL, aug 2005.

[17] 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_J_C, volume 54, number 8, pages 1025 - 1040, aug 2005.

[18] F. Moraes, N. Calazans, A. Mello, L. Möller, L. Ost: *HERMES: an infrastructure for low area overhead packet-switching networks on chip*, Integration, the VLSI Journal, Volume 38, Issue 1, Pages 69-93, October 2004.

[19] L. M. Ni, P. K. McKinley: *A Survey of Wormhole Routing Techniques in Direct Networks*, IEEE Computer Society Press, Volume 26, Issue 2, Pages 62-76, February 1993.

[20] M. Majer, C. Bobda, A. Ahmadinia and J. Teich: *Packet Routing in Dynamically Changing Networks on Chip*, Proc. of 19th IEEE International Parallel and Distributed Symposium, apr 2005.

[21] I. Sobel, G. Feldman: A 3x3 Isotropic Gradient Operator for Image Processing, presented at a talk at the Stanford Artificial Project in 1968

[22] W. Luk, N. Shirazi, P.Y.K. Cheung: *Modeling and Optimizing Run-time Reconfiguration Systems*, FPGAs for Custom Computing Machines, 1996. Proceedings. IEEE Symposium on, pages 167-176, Napa, CA, USA.

[23] S. McMillan and S. Guccione: *Partial Run-Time Reconfiguration Using JRTR*, Proceedings of the Roadmap to Reconfigurable Computing, 10th International Workshop on Field-Programmable Logic and Applications, Springer-Verlag, Lecture Notes in Computer Science, Vol. 1896, Pages: 352 - 360, 2000.

[24] J. Duato, S. Yalamanchili and L. Ni: *Interconnection Networks: an Engineering Approach*, Morgan Kaufmann Publishers Inc., 2003.

[25] C. Bobda and A. Ahmadinia: *Dynamic Interconnection of reconfigurable modules on reconfigurable devices*, Design & Test of Computers, IEEE, volume 22, issue 5, pages 443-451, sep 2005.

[26] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. van Meergergen, P. Wielage and E. Waterlander: *Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip*, Proceedings of the conference on Design, Automation and Test in Europe, March 3-7, 2003.

[27] E. Rijpkema, K. G. W. Goossens and P. Wielage: *A Router Architecture for networks on silicon*, in 2nd Workshop on Embedded Systems (PROGRESS'01), November 2001, pp. 181 - 188.

[28] Open Core Protocol Specification, Release 2.1, OCP-IP, 2005, http://www.ocpip.org/socket/ocpspec/